

IEEE Standard for Actuator Interface for Cyber and Physical Worlds

IEEE Computer Society

Developed by the
Standards Activities Board (SAB) Standard Committee

IEEE Std 2888.2™-2023

STANDARDS

IEEE Standard for Actuator 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 controlling the actuators, which helps enable the definition of interfaces between the cyber world and the physical world, are defined in this standard, including:

- Data formats for interfacing actuator commands defining a root schema with individual actuators
- Data formats for recognizing actuator capabilities defining a root schema with individual actuators
- Data formats for common data types
- Application programming interfaces to transmit commands and capability descriptions of the actuators between the cyber and physical worlds

Keywords: actuator application programming interfaces (APIs), actuator capabilities, actuator data formats, IEEE 2888.2™, large space virtual reality (VR) training system actuators

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 9 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 979-8-8557-0299-6 STD26604
Print: ISBN 979-8-8557-0300-9 STDPD26604

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 balloting 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

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. 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*
SangKwon Jeong, *Secretary*
Tae-Boem Lim, *Task Group Chair*
Shin Kim, *Technical Editor*

Hyun-Soek Ahn
Eunji Choi
Sungwook Kang

Young-Han Kim
Tai-Gil Kwon
Gookhwan Lee

Yegi Lee
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
Galina Antonova
Myung-Sun Baek
Pin Chang
JeongHwoan Choi
Donald Dunn
Werner Hoelzl
Wonki Hong
SangKwon Jeong

Piotr Karocki
Sang-Kyun Kim
Shin Kim
Tai-Gil Kwon
KangYoon Lee
Seung Wook Lee
Greg Luri
HyeonWoo Nam

Bansi Patel
George Percivall
Venkatesha Prasad
Kwanghyun Ro
Suhhee Yoo
Changseok Yoon
Kyoungro Yoon
Yu Yuan
Janusz Zalewski

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 2888.2-2023, IEEE Standard for Actuator Interface for Cyber and Physical Worlds.

This standard defines the vocabulary, data formats, and application programming interfaces (APIs) for controlling the actuators, which helps enable 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 actuators	12
4.1 Root schema structure.....	12
4.2 Data formats for individual actuator.....	21
5. Data formats for describing actuator capabilities	47
5.1 Actuator capability base data.....	47
5.2 Data formats for individual actuator capabilities.....	51
6. Common data type.....	80
6.1 General	80
6.2 Unit type	80
6.3 Color type	83
6.4 Scent type	105
6.5 Sprayer type.....	120
6.6 Unmanned moving platform type	120
6.7 Float3DVector type	121
6.8 FloatArray type.....	122
7. Application programming interfaces for actuators	122
7.1 General	122
7.2 Application programming interface for an individual actuator.....	123
Annex A (informative) Full schema of data format for an interfacing actuator	135
Annex B (informative) Full schema of data format for describing actuator capabilities.....	170

IEEE Standard for Actuator Interface for Cyber and Physical Worlds

1. Overview

1.1 Scope

This standard defines the vocabulary, requirements, metrics, data formats and application program interfaces (APIs) for describing characteristics of, setting up parameters for, and commanding actuators enabling definition of interfaces between the cyber world and the physical world. These actuators shall be defined in both the 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 21778:2017(en) Information technology—The JSON data interchange syntax.³

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

actuator: A device that shall control other devices or affect its environments.

actuator capability: Description of the actuator characteristics such as maximum intensity and minimum frequency.

actuator command: Data commanded for control or activation of an actuator.

3.2 Acronyms and abbreviations

3D	three dimensions
APIs	application programming interfaces
ERM	eccentric rotating mass
GPS	global positioning system
HSV	hue saturation value
ID	identifier
JSON	Javascript object notation
LRA	linear resonant actuator
RGB	red, green, blue
RPM	rotations per minute

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.

SI	Système International d'Unités or International System of Units
UAV	unmanned aerial vehicle
URI	uniform resource identifier
UGV	unmanned ground vehicle
USV	unmanned surface vehicle
UUV	unmanned underwater vehicle

4. Data formats for interfacing actuators

4.1 Root schema structure

4.1.1 General

This clause defines the root schema structure for the interfacing actuator command. This root schema structure specifies tools for describing actions that each actuator is supposed to take. The syntax of the command shall be defined according to ISO/IEC 21778:2017(en).⁵

4.1.2 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "actuator command",
  "description": "Schema for command",
  "type": "object",
  "properties": {
    "timeStamp": {
      "type": "string",
      "format": "date-time"
    },
    "referenceTime": {
      "type": "string",
      "format": "date-time"
    }
  }
}

// This is where the properties of each type of command data are
// declared by reference.
```

⁵ Information on references can be found in Clause 2.

```
// For Example,  
//   "lightCommand": {  
//     "$ref": "#/definitions/lightCommand"  
//   },  
//   "flashCommand": {  
//     "$ref": "#/definitions/flashCommand"  
//   },  
//   ...  
// },  
"additionalProperties": false,  
"minProperties": 1,  
"maxProperties": 3,  
"definitions": {  
  "commandInfoBaseAttributes": {  
    "additionalProperties": false,  
    "type": "object",  
    "properties": {  
      "id": {  
        "type": "string"  
      },  
      "actuationTime": {  
        "anyOf": [  
          {  
            "type": "string",  
            "format": "date-time"  
          },  
          {  
            "type": "string"  
          }  
        ]  
      },  
      "actuatorIdRef": {  
        "type": "string"  
      },  
      "activate": {  
        "type": "boolean"  
      }  
    }  
  },  
}
```

```
"required": ["actuatorIdRef"]
},

"genericCommand": {
  "genericIntensityCommand": {
    "type": "object",
    "properties": {
      "unit": {
        "$ref": "#/definitions/unitType"
      }
    },
  },
  "anyOf": [
    {
      "if": {
        "properties": {
          "unit": {
            "const": "percentage"
          }
        }
      },
      "then": {
        "properties": {
          "value": {
            "type": "number",
            "minimum": 0,
            "maximum": 100
          }
        }
      },
      "else": {
        "properties": {
          "value": {
            "type": "number"
          }
        }
      }
    },
    {

```

```
      "if": {
        "properties": {
          "unit": {
            "not": {}
          }
        },
        "then": {
          "properties": {
            "value": {
              "type": "number"
            }
          }
        },
        "else": {
          "properties": {
            "value": {
              "type": "number",
              "minimum": 0,
              "maximum": 100
            }
          }
        }
      },
      "required": [
        "value"
      ]
    },
    "genericFrequencyCommand": {
      "type": "object",
      "properties": {
        "unit": {
          "$ref": "#/definitions/unitType"
        }
      },
      "anyOf": [
        {
```

```
"if": {
  "properties": {
    "unit": {
      "const": "percentage"
    }
  }
},
"then": {
  "properties": {
    "value": {
      "type": "number",
      "minimum": 0,
      "maximum": 100
    }
  }
},
"else": {
  "properties": {
    "value": {
      "type": "number"
    }
  }
},
{
  "if": {
    "properties": {
      "unit": {
        "not": {}
      }
    }
  },
  "then": {
    "properties": {
      "value": {
        "type": "number"
      }
    }
  }
}
```

```
    },  
    "else": {  
      "properties": {  
        "value": {  
          "type": "number",  
          "minimum": 0,  
          "maximum": 100  
        }  
      }  
    }  
  },  
  "required": [  
    "value"  
  ]  
}  
  
// This is where the properties of each type of command are defined.  
// For Example,  
//   "lightCommand": {  
//     ...  
//   },  
//   "flashCommand": {  
//     ...  
//   },  
//   ...  
}
```

4.1.3 Semantics

The semantics of the root element are shown in Table 1.

Table 1—Semantics of the root element

Name	Definition
commandInfoBaseAttributes	It describes a group of attributes for the command that each actuator is supposed to take.
timeStamp	It describes the creation time of a given metadata.
referenceTime	It describes the time that shall be referenced within the scope of a given metadata.
id	It describes the unique identifier of an individual command.

Name	Definition
actuationTime	It specifies the time when the given actuation command shall be activated. If <code>actuationTime</code> is given in date-time format, it is an absolute time, and if given in string format, it specifies time offset from <code>referenceTime</code> . For example, it is an absolute time if the <code>actuationTime</code> is given in date-time format such as 2022-10-20T14:49:39+07:00. Or if the <code>actuationTime</code> is given in string such as P10M, it is a relative time. Regular expression for relative time is P{1}(0-9)+(S M H){1}.
actuatorIdRef	It describes a link to the actuator that generated the data contained in this particular instance.
activate	It tells the command to turn on and off the device or actuator. A value of “true” means the actuator is commanded to activate, and a value of “false” means the actuator is commanded to deactivate. In binary representation, a value of “1” means the actuator is commanded to activate, and a value of “0” means the actuator is commanded to deactivate.
genericCommand	It describes the command using <code>genericIntensityCommand</code> and <code>genericFrequencyCommand</code> .
genericIntensityCommand	It provides the intensity value that the actuator shall produce. The value format shall be set as a percentage or an absolute value that the actuator shall have. If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.
genericFrequencyCommand	It provides the frequency value that the actuator shall produce. The value format shall be set as a percentage or an absolute value that the actuator shall have. If the frequency unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.1.4 Examples

In the following example, a light actuator commands the intensity of 130 lux with the color “blue” with the timestamp, which means the creation time of the metadata. This light actuator is active. The command `id` of this example is “C00000001”, and the `actuatorIdRef` is “DC-01”.

```
{
  "timeStamp": "2022-11-13T18:47:09+00:00",
  "lightCommand": {
    "commandInfoBaseAttributes": {
      "id": "C00000001",
      "actuatorIdRef": "DC-01",
      "activate": true
    },
    "color": "blue",
    "intensity": {
      "value": 130
    }
  }
}
```

In the following example, a light actuator commands the intensity of 100 lux with the color “white” with the timestamp, which means the time when the actuator data command is made. This light actuator is active. The command `id` of this example is “C00000001”, and the `actuatorIDRef` is “DC-05”. The time that the actuator is referencing is 20:20:39(UTC), 20 October 2022, and the relative time is 10 min, which means that the command shall be operated 10 min past the reference time.

```
{
  "referenceTime": "2022-10-20T20:20:39+00:00",
  "lightCommand": {
    "commandInfoBaseAttributes": {
      "id": "C00000001",
      "actuationTime": "P10M",
      "actuatorIdRef": "DC-05",
      "activate": true
    },
    "color": "white",
    "intensity": {
      "value": 100
    }
  }
}
```

In this example, a scent actuator (DC-24) commands the intensity of 10% with the scent “coconut” with the timestamp, which means the creation time of the metadata. The command `id` of this example is “C00000001”, and the `actuatorIDRef` is “DC-24”. The command shall be operated at 20:00:00(UTC+9), 14 November 2022.

```
{
  "timeStamp": "2022-11-13T18:47:09+00:00",
  "scentCommand": {
    "commandInfoBaseAttributes": {
      "id": "C00000010",
      "actuatorIdRef": "DC-24",
      "actuationTime": "2022-11-14:T20:00:00:09+00"
    },
    "scent": "coconut",
    "intensity": {
      "unit": "percentage",
      "value": 10
    }
  }
}
```

In this example, an ERM vibration actuator (DC-50) commands the intensity of 50% with the timestamp that indicates the creation time of given metadata. The command `id` of this example is “C00000001”, and the `actuatorIDRef` is “DC-50”. The command shall be operated 30 s past reference time (18:48:50(UTC), 13 November 2022).

```
{
  "timeStamp": "2022-11-13T18:47:09+00:00",
  "referenceTime": "2022-11-13T18:48:50+00:00",
  "ERMVibrationCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-50",
      "actuationTime": "P30S"
    },
    "intensity": {
      "unit": "percentage",
      "value": 50
    }
  }
}
```

This example shows the description of the actuator command of light effect. This light actuator (DC-03) is commanded to turn on the red color (Red:255, Green:0, Blue:0) light and, the command should be operated at 09:00:00(UTC+5), 6 September 2023.

```
{
  "lightCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-03",
      "actuationTime": "2024-09-06T09:00:00+05:00"
    },
    "color": {
      "Red": 255,
      "Green": 0,
      "Blue": 0
    }
  }
}
```

4.2 Data formats for individual actuator

4.2.1 General

This subclause specifies individual actuator command types. The actuators defined in this standard include the following:

- Light actuator
- Flash actuator
- Scent actuator
- Heating actuator
- Cooling actuator
- Piezo vibration actuator
- LRA vibration actuator
- ERM vibration actuator
- Generic vibration actuator
- Sprayer actuator
- Fog actuator
- Wind actuator
- Bubble actuator
- Step motor actuator
- Character display actuator
- Unmanned moving platform actuator

4.2.2 Light actuator

4.2.2.1 General

This subclause specifies an actuator command type that shall generate a light effect.

4.2.2.2 Syntax

```
"lightCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "color": {  
      "$ref": "#/definitions/colorType"  
    }  
  }  
}
```

```

    },
    "intensity": {
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"
    }
  },
  "additionalProperties": false
},

```

4.2.2.3 Semantics

The semantics of the `lightCommand` are shown in Table 2.

Table 2—Semantics of `lightCommand`

Name	Definition
<code>lightCommand</code>	Tool for describing a command for a light actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>color</code>	It describes the color that the light actuator shall provide as a reference to a term that shall be using the <code>colorType</code> defined in 6.3.
<code>intensity</code>	It describes the intensity value that a light actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.2.4 Examples

This example shows the description of the actuator command of the light effect. This light actuator (DC-01) is commanded to perform the intensity of 130 lux with the color “blue,” and the command is set to be operated at 09:00:00(UTC+5), 1 November 2022.

```

{
  "lightCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-01",
      "actuationTime": "2022-11-01T09:00:00+05:00"
    },
    "color": "blue",
    "intensity": {
      "unit": "lux",
      "value": 130
    }
  }
}

```

4.2.3 Flash actuator

4.2.3.1 General

This subclause specifies an actuator command type that shall generate a flash effect.

4.2.3.2 Syntax

```
"flashCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "color": {  
      "$ref": "#/definitions/colorType"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    },  
    "frequency": {  
      "$ref": "#/definitions/genericCommand/genericFrequencyCommand"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.3.3 Semantics

The semantics of the `flashCommand` are shown in Table 3.

Table 3—Semantics of the flashCommand

Name	Definition
<code>flashCommand</code>	Tool for describing a command for a flash actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>color</code>	It describes the color that the flash actuator shall provide as a reference to a term that shall be using the <code>colorType</code> defined in 6.3.
<code>intensity</code>	It describes the intensity value that a flash actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is "percentage," it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

Name	Definition
frequency	It describes the frequency value that a flash actuator shall generate. The frequency value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericFrequencyCommand</code> . If the frequency unit is "percentage", it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.3.4 Examples

This example shows the description of the actuator command for the flash effect. This flash actuator (DC-02) is commanded to set the light intensity at 80 lux and the frequency at 120 Hz.

```
{
  "flashCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-02"
    },
    "color": "blue",
    "intensity": {
      "unit": "lux",
      "value": 80
    },
    "frequency": {
      "unit": "hz",
      "value": 120
    }
  }
}
```

4.2.4 Scent actuator

4.2.4.1 General

This subclause specifies an actuator command type that shall generate a scent effect.

4.2.4.2 Syntax

```
"scentCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
```

```
    },  
    "scent": {  
      "$ref": "#/definitions/scentType"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false,  
  "required": [  
    "scent"  
  ]  
},
```

4.2.4.3 Semantics

The semantics of the `scentCommand` are shown in Table 4.

Table 4—Semantics of the `scentCommand`

Name	Definition
<code>scentCommand</code>	Tool for describing a command for a scent actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>scent</code>	It describes the scent that a specific scent actuator shall generate. The scent unit of the command value describes as a reference to a term that shall be using the <code>scentType</code> defined in 6.4.
<code>intensity</code>	It describes the intensity value that a scent actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.4.4 Examples

This example shows the description of an actuator command (C00000010) for the scent effect. This scent actuator (DC-13) is commanded to set the intensity to 5% of the maximum intensity with the scent “rose.”

```
{  
  "scentCommand": {  
    "commandInfoBaseAttributes": {  
      "id": "C00000010",  
      "actuatorIdRef": "DC-13"  
    },  
    "scent": "rose",  
  }  
}
```

```
"intensity": {  
  "unit": "percentage",  
  "value": 5  
}  
}  
}
```

4.2.5 Heating actuator

4.2.5.1 General

This subclause specifies an actuator command type that shall generate a heating effect.

4.2.5.2 Syntax

```
"heatingCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.5.3 Semantics

The semantics of the `heatingCommand` are shown in Table 5.

Table 5—Semantics of the `heatingCommand`

Name	Definition
<code>heatingCommand</code>	Tool for describing a command for a heating actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>intensity</code>	It describes the intensity value that a heating actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.5.4 Examples

This example shows the description of an actuator command (C00000015) for the heating effect. This heating actuator (DC-03) is commanded to set the intensity to 42 °C, which is the maximum intensity.

```
{
  "heatingCommand": {
    "commandInfoBaseAttributes": {
      "id": "C00000015",
      "actuatorIdRef": "DC-03"
    },
    "intensity": {
      "unit": "celsius",
      "value": 42
    }
  }
}
```

4.2.6 Cooling actuator

4.2.6.1 General

This subclause specifies an actuator command type that shall generate a cooling effect.

4.2.6.2 Syntax

```
"coolingCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
    },
    "intensity": {
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"
    }
  },
  "additionalProperties": false
},
```

4.2.6.3 Semantics

The semantics of the `coolingCommand` are shown in Table 6.

Table 6— Semantics of the coolingCommand

Name	Definition
coolingCommand	Tool for describing a command for a cooling actuator.
commandInfoBaseAttributes	It describes a group of attributes for the command that each actuator is supposed to take.
intensity	It describes the intensity value that a cooling actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.6.4 Examples

This example shows the description of an actuator command (C00000020) for the cooling effect. This cooling actuator (DC-04) is commanded to set the temperature to -2°C .

```
{
  "coolingCommand": {
    "commandInfoBaseAttributes": {
      "id": "C00000020",
      "actuatorIdRef": "DC-04"
    },

    "intensity": {
      "unit": "celsius",
      "value": -2
    }
  }
}
```

4.2.7 Piezo vibration actuator

4.2.7.1 General

This subclause specifies an actuator command type that shall generate a piezo vibration effect.

4.2.7.2 Syntax

```
"piezoVibrationCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
```

```
    },  
    "synthesisModeArray": {  
      "type": "array",  
      "$ref":  
"#/definitions/piezoVibrationCommand/properties/synthesisMode"  
    },  
    "synthesisMode": {  
      "items": {  
        "type": "object",  
        "properties": {  
          "amplitude": {  
            "type": "integer",  
            "minimum": 0  
          },  
          "frequency": {  
            "type": "integer",  
            "minimum": 0  
          },  
          "duration": {  
            "type": "integer",  
            "minimum": 0  
          }  
        },  
        "additionalProperties": false,  
        "uniqueItems": true  
      },  
      "minItems": 1  
    },  
    "additionalProperties": false  
  }  
}
```

4.2.7.3 Semantics

The semantics of the `piezoVibrationCommand` are shown in Table 7.

Table 7—Semantics of the piezoVibrationCommand

Name	Definition
piezoVibrationCommand	Tool for describing a command for a piezo vibration.
synthesisModeArray	It describes an array of synthesis modes used to create a wide variety of effects.
synthesisMode	It describes a synthesis mode used to create a synthesized sinusoid, which shall be generated by manipulating the parameters of amplitude, frequency, and duration.
amplitude	It describes the amplitude of the synthesized sinusoid.
frequency	It describes the frequency of the synthesized sinusoid.
duration	It describes the duration (number of cycles) to be played by the synthesizer.

4.2.7.4 Examples

This example shows the description of an actuator command of the piezo vibration effect with the following semantics. This piezo vibration actuator (DC-23) is commanded to perform based on three effects (amplitude, frequency, and duration) in the `synthesisMode`. After finishing effects of {amplitude: 255, frequency: 15, duration: 50}, the next effects shall be performed.

```
{
  "piezoVibrationCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-23"
    },
    "synthesisMode": [
      {
        "amplitude": 255,
        "frequency": 15,
        "duration": 50
      },
      {
        "amplitude": 255,
        "frequency": 17,
        "duration": 50
      }
    ]
  }
}
```

4.2.8 Linear resonance actuator vibration actuator

4.2.8.1 General

This subclause specifies an actuator command type that shall generate an LRA vibration effect.

4.2.8.2 Syntax

```
"LRAVibrationCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.8.3 Semantics

The semantics of the LRAVibrationCommand are shown in Table 8.

Table 8—Semantics of the LRAVibrationCommand

Name	Definition
LRAVibrationCommand	Tool for describing a command for a LRA vibration actuator.
intensity	It describes the intensity value that a vibration actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using genericIntensityCommand. If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.8.4 Examples

This example shows the description of an actuator command for the vibration effect with the following semantics. The LRA vibration actuator (DC-31) is commanded to set the intensity to 40% of the maximum intensity.

```
{  
  "LRAVibrationCommand": {  
    "commandInfoBaseAttributes": {  
      "actuatorIdRef": "DC-31"  
    }  
  }  
}
```

```
    },  
    "intensity": {  
      "unit": "percentage",  
      "value": 40  
    }  
  }  
}
```

4.2.9 Eccentric rotation mass vibration actuator

4.2.9.1 General

This subclause specifies an actuator command type that shall generate an ERM vibration effect.

4.2.9.2 Syntax

```
"ERMVibrationCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.9.3 Semantics

The semantics of the `ERMVibrationCommand` are listed in Table 9.

Table 9—Semantics of the `ERMVibrationCommand`

Name	Definition
<code>ERMVibrationCommand</code>	Tool for describing a command for an ERM vibration actuator.
<code>intensity</code>	It describes the intensity value that a vibration actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.9.4 Examples

This example shows the description of an actuator command for the vibration effect with the following semantics. This ERM vibration actuator (DC-31) is commanded to set the intensity to 50% of the maximum intensity.

```
{
  "ERMVibrationCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-31"
    },
    "intensity": {
      "unit": "percentage",
      "value": 50
    }
  }
}
```

4.2.10 Generic vibration actuator

4.2.10.1 General

This subclause specifies an actuator command type that shall generate a generic vibration effect.

4.2.10.2 Syntax

```
"genericVibrationCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
    },
    "intensity": {
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"
    },
    "frequency": {
      "$ref": "#/definitions/genericCommand/genericFrequencyCommand"
    }
  },
  "additionalProperties": false,
  "anyOf": [
```

<pre>{ "required": ["intensity"], { "required": ["frequency"] } }</pre>

4.2.10.3 Semantics

The semantics of the `genericVibrationCommand` are shown in Table 10.

Table 10—Semantics of `genericVibrationCommand`

Name	Definition
<code>genericVibrationCommand</code>	Tool for describing a command for a generic vibration actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>intensity</code>	It describes the intensity value that a generic vibration actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.
<code>frequency</code>	It describes the frequency value that a generic vibration actuator shall generate. The frequency value shall be set as either the percentage value for the maximum value or the specific actuator value that the actuator shall produce using <code>genericFrequencyCommand</code> . If the frequency unit is “percentage,” it is set to a percentage value; otherwise, it is set to an absolute value coming from individual actuator capabilities.

4.2.10.4 Examples

This example shows the description of an actuator command for a generic vibration effect. The generic vibration actuator (DC-05) is commanded to set the intensity to 40% of the maximum intensity and the frequency to 20% of the maximum frequency.

<pre>{ "genericVibrationCommand": { "commandInfoBaseAttributes": { "actuatorIdRef": "DC-05" }, </pre>

```
"intensity": {  
  "unit": "percentage",  
  "value": 40  
},  
"frequency": {  
  "unit": "percentage",  
  "value": 20  
}  
}  
}
```

4.2.11 Sprayer actuator

4.2.11.1 General

This subclause specifies an actuator command type that shall generate a spraying effect.

4.2.11.2 Syntax

```
"sprayerCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "sprayingType": {  
      "$ref": "#/definitions/sprayingType"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false,  
  "required": [  
    "sprayingType"  
  ]  
},
```

4.2.11.3 Semantics

The semantics of the `sprayerCommand` are shown in Table 11.

Table 11—Semantics of the `sprayerCommand`

Name	Definition
<code>sprayerCommand</code>	Tool for describing a command for a <code>sprayer</code> actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>sprayingType</code>	It describes the type of the sprayed material that shall be using the <code>sprayingType</code> defined in 6.4.
<code>intensity</code>	It describes the intensity value that a <code>sprayer</code> actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or as the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set as an absolute value coming from individual actuator capabilities.

4.2.11.4 Examples

This example shows the description of an actuator command for the `sprayer` effect. The `sprayer` actuator (DC-06) is commanded to spray pure water, and the intensity is set to 45% of the maximum intensity.

```
{
  "sprayerCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-06"
    },
    "sprayingType": "water",
    "intensity": {
      "unit": "percentage",
      "value": 45
    }
  }
}
```

4.2.12 Fog actuator

4.2.12.1 General

This subclause specifies an actuator command type that shall generate a fog effect.

4.2.12.2 Syntax

```
"fogCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.12.3 Semantics

The semantics of the `fogCommand` are shown in Table 12.

Table 12—Semantics of fogCommand

Name	Definition
<code>fogCommand</code>	Tool for describing a command for a fog actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>intensity</code>	It describes the intensity value that a fog actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or as the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set as an absolute value coming from individual actuator capabilities.

4.2.12.4 Examples

This example shows the description of an actuator command for the fog effect. The fog actuator (DC-07) is commanded to set the intensity to 50% of the maximum intensity.

```
{  
  "fogCommand": {  
    "commandInfoBaseAttributes": {  
      "actuatorIdRef": "DC-07"  
    },  
    "intensity": {  
      "unit": "percentage",  
      "value": 50  
    }  
  }  
}
```

```
}  
  
}  
  
}
```

4.2.13 Wind actuator

4.2.13.1 General

This subclause specifies an actuator command type that shall generate a wind effect.

4.2.13.2 Syntax

```
"windCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "intensity": {  
      "$ref": "#/definitions/genericCommand/genericIntensityCommand"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.13.3 Semantics

The semantics of the `windCommand` are shown in Table 13.

Table 13—Semantics of `windCommand`

Name	Definition
<code>windCommand</code>	Tool for describing a command for a wind actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>intensity</code>	It describes the intensity value that a wind actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or as the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is “percentage,” it is set to a percentage value; otherwise, it is set as an absolute value coming from individual actuator capabilities.

4.2.13.4 Examples

This example shows the description of an actuator command for the wind effect. The wind actuator (DC-08) is commanded to set the intensity to 60% of the maximum intensity.

```
{
  "windCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-08"
    },
    "intensity": {
      "unit": "percentage",
      "value": 60
    }
  }
}
```

4.2.14 Bubble actuator

4.2.14.1 General

This subclause specifies an actuator command type that shall generate a bubble effect.

4.2.14.2 Syntax

```
"bubbleCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
    }
  },
  "additionalProperties": false
},
```

4.2.14.3 Semantics

The semantics of the `bubbleCommand` are shown in Table 14.

Table 14—Semantics of `bubbleCommand`

Name	Definition
<code>bubbleCommand</code>	Tool for describing a command for a bubble actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.

Name	Definition
intensity	It describes the intensity value that a bubble actuator shall generate. The intensity value shall be set as either the percentage value for the maximum value or as the specific actuator value that the actuator shall produce using <code>genericIntensityCommand</code> . If the intensity unit is "percentage," it is set to a percentage value; otherwise, it is set as an absolute value coming from individual actuator capabilities.

4.2.14.4 Examples

This example shows the description of an actuator command for the bubble effect. The bubble actuator (DC-20) is commanded to set the intensity to 100% of the maximum intensity.

```
{
  "bubbleCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-20"
    },
    "intensity": {
      "unit": "percentage",
      "value": 100
    }
  }
}
```

4.2.15 Step motor actuator

4.2.15.1 General

This subclause specifies an actuator command type that shall control a step motor.

4.2.15.2 Syntax

```
"stepMotorCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
    },
    "speed": {
      "type": "integer",
      "minimum": 0
    }
  }
}
```

```

    },
    "steps": {
      "type": "integer",
      "minimum": 0
    },
    "orientation": {
      "type": "integer",
      "enum": [
        1,
        -1
      ]
    }
  },
  "additionalProperties": false
},

```

4.2.15.3 Semantics

The semantics of the `stepMotorCommand` are shown in Table 15.

Table 15—Semantics of the `stepMotorCommand`

Name	Definition
<code>stepMotorCommand</code>	Tool for describing a command for a step motor actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>speed</code>	It describes the speed at which the step motor actuator shall rotate in RPM, which is the unit of measurement for the specific actuator's rotational capabilities.
<code>steps</code>	It describes the number of steps that the step motor actuator shall rotate through a given angle. For example, when the fixed step angle of the step motor is 5.625°, and the fixed speed variation ratio is 0.015625(1/64), the number of steps for one revolution is $(360 / 5.265) \times 64 = 4096$. For example, when the fixed step angle of the step motor is 5.625°, and the fixed speed variation ratio is 0.015625(1/64), setting the steps to 2,048 means the stepper motor is commanded to perform a half-revolution rotation.
<code>orientation</code>	It describes the orientation that the step motor actuator shall indicate the direction of rotation. If the orientation is 1, it means that the stepper motor rotates counterclockwise, and -1 means that it rotates clockwise.

4.2.15.4 Examples

This example shows the description of the actuator command for the step motor. The step motor actuator (DC-21) is commanded to rotate clockwise with a speed of 10 RPM, performing 32 steps.

```
{  
  "stepMotorCommand": {  
    "commandInfoBaseAttributes": {  
      "actuatorIdRef": "DC-21"  
    },  
    "speed": 10,  
    "steps": 32,  
    "orientation": -1  
  }  
}
```

4.2.16 Character display actuator

4.2.16.1 General

This subclause specifies an actuator command type that shall control a character display actuator.

4.2.16.2 Syntax

```
"characterDisplayCommand": {  
  "type": "object",  
  "properties": {  
    "commandInfoBaseAttributes": {  
      "$ref": "#/definitions/commandInfoBaseAttributes"  
    },  
    "displayMode": {  
      "type": "string",  
      "enum": [  
        "text_mode",  
        "graphics_mode"  
      ]  
    },  
    "xCoordinate": {  
      "type": "integer",  
      "minimum": 0  
    },  
    "yCoordinate": {
```

```
        "type": "integer",
        "minimum": 0
    },
    "foregroundColor": {
        "$ref": "#/definitions/colorType"
    },
    "backgroundColor": {
        "$ref": "#/definitions/colorType"
    },
    "textSize": {
        "type": "string",
        "pattern": "^[0-9]+x[0-9]+$"
    },
    "text": {
        "type": "string"
    }
},
"required": [
    "xCoordinate",
    "yCoordinate",
    "text"
],
"additionalProperties": false,
}
```

4.2.16.3 Semantics

The semantics of the `characterDisplayCommand` are shown in Table 16.

Table 16—Semantics of the `characterDisplayCommand`

Name	Definition
<code>characterDisplayCommand</code>	Tool for describing a command for a character display actuator.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>displayMode</code>	It describes the type of display mode supported by a character display actuator, which is classified into <code>text_mode</code> and <code>graphics_mode</code> . <code>text_mode</code> is a display mode divided into rows and columns of boxes showing characters. The font size, foreground color, and background color for characters are fixed in <code>text_mode</code> . <code>graphics_mode</code> is a display mode divided into pixels vertically and horizontally. The font size, foreground color, and background color for characters are variable in <code>graphics_mode</code> .

Name	Definition
xCoordinate	It describes the <i>x</i> -coordinate of text position on a two-dimensional screen. In <i>text_mode</i> , it represents the column number of the first character in a text string. In <i>graphics_mode</i> , it represents the <i>x</i> -coordinate of the top-left corner pixel of the first character bounding box in a text string. The minimum <i>x</i> -coordinate is zero.
yCoordinate	It describes the <i>y</i> -coordinate of text position on a two-dimensional screen. In <i>text_mode</i> , it represents the row number of first character in a text string is displayed. In <i>graphics_mode</i> , it represents the <i>y</i> -coordinate of the top-left corner pixel of the first character bounding box in a text string. The minimum <i>y</i> -coordinate is zero.
Foreground	It describes a text foreground color. If omitted, the current foreground color is used. If the type of display mode is <i>text_mode</i> , it is ignored.
Background	It describes a text background color. If omitted, the background color is treated as a transparent color. If the type of display mode is <i>text_mode</i> , it is ignored.
textSize	It describes the font size of the text to be printed on the screen. The font size is represented into (number of horizontal pixels) × (number of vertical pixels). If omitted, the current text size is used. If the type of display mode is <i>text_mode</i> , it is ignored.
Text	It describes a text string including letters, numbers, symbols, and simple figures that shall be printed on a screen.

4.2.16.4 Examples

This example shows the description of the actuator command for character display with the following semantics. The character display actuator (DC-23) that supports *graphics_mode* outputs the text “hello world!” with a red foreground color, black background color, and text size 12x16 on the screen at (12, 5) coordinates.

```
{
  "characterDisplayCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-23"
    },
    "displayMode": "graphics_mode",
    "xCoordinate": 12,
    "yCoordinate": 5,
    "foregroundColor": "red",
    "backgroundColor": "black",
    "textSize": "12x16",
    "text": "hello world!"
  }
}
```

4.2.17 Unmanned moving platform actuator

4.2.17.1 General

This subclause specifies an actuator command type that shall control an unmanned moving platform actuator.

4.2.17.2 Syntax

```
"unmannedMovingPlatformCommand": {
  "type": "object",
  "properties": {
    "commandInfoBaseAttributes": {
      "$ref": "#/definitions/commandInfoBaseAttributes"
    },
    "unmannedMovingPlatform": {
      "$ref": "#/definitions/unmannedMovingPlatformType"
    },
    "sequenceNumber": {
      "type": "integer",
      "minimum": 0
    },
    "duration": {
      "type": "number"
    },
    "linearVelocity": {
      "$ref": "#/definitions/float3DVectorType"
    },
    "linearVelocityUnit": {
      "$ref": "#/definitions/unitType",
      "default": "meterpersec"
    },
    "angularVelocity": {
      "$ref": "#/definitions/float3DVectorType"
    },
    "angularVelocityUnit": {
      "$ref": "#/definitions/unitType",
      "default": "radianpersec"
    }
  }
}
```

```

    },
    "additionalProperties": false,
    "required": [
        "unmannedMovingPlatform"
    ]
}

```

4.2.17.3 Semantics

The semantics of the `unmannedMovingPlatformCommand` are shown in Table 17.

Table 17—Semantics of the `unmannedMovingPlatformCommand`

Name	Definition
<code>unmannedMovingPlatformCommand</code>	Tool for describing a command for unmanned moving platform.
<code>commandInfoBaseAttributes</code>	It describes a group of attributes for the command that each actuator is supposed to take.
<code>unmannedMovingPlatform</code>	It describes the type of the unmanned moving platform that shall be using the <code>unmannedMovingPlatformType</code> defined in 6.6
<code>sequenceNumber</code>	It describes sequence number incremented by one that helps to guarantee the order of data transmission.
<code>duration</code>	It describes how many seconds to maintain the velocity. If 0, maintain the speed until the next command is issued.
<code>linearVelocity</code>	It describes the velocity in linear portion of the free space. The three numbers in turn refer to the front (<i>x</i> -axis), the left (<i>y</i> -axis), and the top (<i>z</i> -axis).
<code>linearVelocityUnit</code>	It describes the linear velocity unit of the command value as a reference to a term that shall be using the <code>unitType</code> defined in 6.2.
<code>angularVelocity</code>	It describes the velocity in angular portion of the free space. The three numbers in turn refer to roll(<i>x</i> -axis), pitch(<i>y</i> -axis), and yaw(<i>z</i> -axis).
<code>angularVelocityUnit</code>	It describes the angular velocity unit of the command value as a reference to a term that shall be using the <code>unitType</code> defined in 6.2.

4.2.17.4 Examples

This example shows a description of the command for an unmanned moving platform with the following semantics. The actuator of the unmanned moving platform (DC-90) moves along the *x*-axis at a linear velocity of 7.2562 m/s and simultaneously rotates clockwise at an angular speed of 3.4856 rad/s for a duration of 3 s along the *z*-axis.

```

{
  "unmannedMovingPlatformCommand": {
    "commandInfoBaseAttributes": {
      "actuatorIdRef": "DC-90"
    },
    "unmannedMovingPlatform": "UGV",
    "duration": 3,
    "linearVelocity": [7.2562, 0.0, 0.0],
  }
}

```

```
"angularVelocity": [0.0, 0.0, 3.4856]
}
}
```

5. Data formats for describing actuator capabilities

5.1 Actuator capability base data

5.1.1 General

This clause defines the root schema structure for interfacing actuator capability. The syntax of command shall be defined according to ISO/IEC 21778:2017(en).

5.1.2 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "actuator capability data",
  "description": "Schema actuator capabilities",
  "type": "object",
  "properties": {
    // This is where the properties of each type of co capability data are
    // declared by reference.
    // For Example,
    //   "lightActuatorCapabilityData": {
    //     "$ref": "#/definitions/lightActuatorCapabilityData"
    //   },
    //   "flashActuatorCapabilityData": {
    //     "$ref": "#/definitions/flashActuatorCapabilityData"
    //   },
    //   ...
  },
  "additionalProperties": false,
  "minProperties": 1,
  "maxProperties": 1,
  "definitions": {
    "actuatorCapabilityBaseData": {
      "type": "object",
```

```
"properties": {
  "id": {
    "type": "string"
  },
  "zerothOrderDelayTime": {
    "type": "number",
    "minimum": 0,
    "default": 0
  },
  "firstOrderDelayTime": {
    "type": "number",
    "minimum": 0,
    "default": 0
  },
  "location": {
    "type": "object",
    "properties": {
      "locationType": {
        "type": "string",
        "enum": [
          "global",
          "local"
        ]
      },
      "x": {
        "type": "number"
      },
      "y": {
        "type": "number"
      },
      "z": {
        "type": "number"
      }
    },
    "required": [
      "locationType",
      "x",
      "y"
    ]
  }
}
```

```

    ]
  }
},
"additionalProperties": false
}
}
}
//This is where the properties of each type of capability data are
defined.
// For Example,
//   "lightActuatorCapabilityData": {
//     ...
//   },
//   "flashActuatorCapabilityData": {
//     ...
//   },
//   ...

```

5.1.3 Semantics

The semantics of the `actuatorCapabilityBaseData` are shown in Table 18.

Table 18—Semantics of the `actuatorCapabilityBaseData`

Name	Definition
<code>actuatorCapabilityBaseData</code>	Tool for describing a group of attributes for the command that each actuator is supposed to take.
<code>id</code>	It describes the unique identifier for identifying a specific actuator. An individual actuator shall be commanded using the data format for the actuator by referencing this ID through the <code>actuatorIdRed</code> field in the command schema.
<code>zerothOrderDelayTime</code>	It describes the required preparation time of an actuator to be activated since it receives a command in the unit of millisecond (ms). If it is not specified, the default value is 0.
<code>firstOrderDelayTime</code>	It describes the delay time for an actuator to reach the target intensity since it receives a command and is activated in the unit of millisecond (ms). If it is not specified, the default value is 0.
<code>location</code>	It describes the location to represent the position of individual actuators. The position of the actuator shall be defined differently depending on the <code>locationType</code>. If <code>locationType</code> is global, the <i>x</i> value means latitude, and the <i>y</i> value means longitude equal to the GPS information, and if <code>locationType</code> is local, the <i>x</i>, <i>y</i>, and <i>z</i> values represent a relative position as a user or system defines the local coordinate system with the origin of the coordinate system.