OGC SensorThings API Tasking Core Discussion Paper

Table of Contents

Publication Date: 2018-12-18

Approval Date: 2018-06-07

Submission Date: 2018-05-17

Reference number of this document: OGC 18-056

Reference URL for this document: http://www.opengis.net/doc/DP/SensorThings-tasking

Category: OGC Discussion Paper

Editor: Steve Liang, Tania Khalafbeigi, Kan Luo

Title: OGC SensorThings API Tasking Core Discussion Paper

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Summary

This discussion paper offers descriptions and provides JSON examples of TaskingCapabilities and Tasks for the SensorThings Application Programming Interface (API).

1.1. Requirements & Research Motivation

The OGC SensorThings API Part II – Tasking Core (OGC 17-079r1) describes the standard and offers the compliance tests specifications. However, more descriptions and examples are needed in order to help users to explore what the SensorThings API (STA) Tasking can do. This discussion paper provides the needed JSON examples for different types of actuators.

1.2. Recommendations for Future Work

This discussion paper is intended to provide recommendations on creating TaskingCapabilities and Tasks in SensorThings. Thus, the recommendations on future work can be found at the end of each section.

1.3. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization
Steve Liang	University of Calgary, Canada / SensorUp Inc.
Tania Khalafbeigi	University of Calgary, Canada / SensorUp Inc.
Kan Luo	University of Calgary, Canada
Hylke van der Schaaf	Fraunhofer, Germany

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC 08-094r1, OGC® SWE Common Data Model, January 2011, http://portal.opengeospatial.org/files/?artifact_id=41157
- OGC 15-078r6, OGC® SensorThings API Part 1: Sensing, July 2016, http://docs.opengeospatial.org/is/15-078r6/15-078r6.html
- OGC 17-079r1, OGC SensorThings API - Part II Tasking Core, Draft, https://portal.opengeospatial.org/files/?artifact_id=78552&version=1
- OGC 17-011r2, OGC® JSON Encoding Rules - SWE Common / SensorML, January 2018, http://docs.opengeospatial.org/bp/17-011r2/17-011r2.html

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- sensor

  ```
  An entity capable of observing a phenomenon and returning an observed value. Type
  of observation procedure that provides the estimated value of an observed property
  at its output. [OGC 12-000]
  ```

- task

  ```
  (Conceptual) resource that represents a SPS assignment. It includes the (possibly
  empty) set of tasking parameters [OGC 09-000].
  ```

- tasking

  ```
  Parameterizing an asset; can be done by sending one or more tasking requests [OGC
  09-000]
  ```

3.1. Abbreviated terms

- API Application Programming Interface
- IoT Internet of Things
- JSON JavaScript Object Notation
- OGC Open Geospatial Consortium
- SensorML Sensor Model Language
- STA SensorThings API
- SWE Sensor Web Enablement

Chapter 4. Overview

Chapter 5 contains the following examples:

Chapter 5. Example Request/Response

The taskingParameters property describes optional and mandatory tasking parameters. Clients use the definition to provide corresponding tasking parameter values. To ensure a common understanding between client and server, a standard exchange protocol is used to express both descriptions and tasking parameter values. SensorThings uses the JSON encoding defined in OGC 17-011r2 to define taskingParameters. However, to be consistency with SensorThings Part 1-Sensing, there are several differences between the OGC SWE common data model and Tasking (Table 1).

Table 1. Differences between SWE Common Data Model and Tasking

	OGC SensorThings API Task Core	OGC SWE Common Data Model
Unit Reference	`"unitOfMeasurement": { "symbol": " °C", "name": "degree Celsius", "definition": "http://unitsofmeasure.org/ucum.html#para-30" }`	**"uom": { "code": "°C " }**
Time Reference Frame and Format	`UTC in ISO format`	**Default frame is UTC, but can be the different time reference system**
Location	`GeoJSON`	**Not supported**

The examples of taskingParameters in TaskingCapabilities and Tasks can be divided into following types according to the parameter types: Boolean, Text, Category, Count, Quantity, Time, Location, CategoryRange, CountRange, QuantityRange, and TimeRange.

5.1. Boolean

Example 1. Turn on/off a light

a) taskingParameters in TaskingCapability

```
{
  "type": "Boolean",
  "name": "status",
  "label": "light on/off status",
  "description": "If set true, the light is on; Otherwise, the light is off"
}
```

b) taskingParameters in Task

```
{
  "status": true
}
```

5.2. Text

Example 2. Set text to display

a) taskingParameters in TaskingCapability

```
{
  "type": "Text",
  "name": "display",
  "label": "display text",
  "description": "Set text to display",
  "constraint": {
    "type": "AllowedTokens",
    "pattern": "^[a-zA-Z ]*$"
  }
}
```

b) taskingParameters in Task

```
{
  "display": "SensorThings API Tasking"
}
```

5.3. Category

Example 3. Select washing machine program

a) taskingParameters in TaskCapability

```
{
  "type": "Category",
  "name": "program",
  "label": "washing machine program",
  "description": "The washing program to use",
  "constraint": {
 "type": "AllowedTokens",
 "value": ["Cotton", "Synthetics", "Wool"]
  }
}
```

b) taskingParameters in Task

```
{
   "program": "Cotton"
}
```

5.4. Count

Example 4. Set the number of times that robotic vacuum will clean the house

a) taskingParameters in TaskingCapability

```
{
   "type": "Count",
   "name": "clean_count",
   "label": "total count to clean the house",
   "description": "The total count of house cleaning",
   "constraint": {
  "type": "AllowedValues",
  "interval": [0 10]
}
}
```

b) taskingParameters in Task

```
{
   "clean_count": 2
}
```

5.5. Quantity

Example 5. Select water temperature for a washing machine

a) taskingParameters in TaskCapability

```
{
  "type": "Quantity",
  "name": "temperature",
  "label": "Washing temperature",
  "description": "The temperature to wash on",
  "unitOfMeasurement": {
    "symbol": " ℃",
    "name": "degree Celsius",
        "definition": "http://unitsofmeasure.org/ucum.html#para-30"
  },
  "constraint": {
    "type": "AllowedValues",
    "value": [30, 40, 60]
  }
}
```

b) taskingParameters in Task

```
{
  "temperature": 30
}
```

Example 6. Control the opening percentage of window

a) taskingParameters in TaskCapability

```
{
   "type": "Quantity",
   "name": "open_percentage",
   "label": "opening percentage",
   "description": "How far the window has to open, in percentage",
   "unitOfMeasurement": {
  "symbol": "%",
  "name": "percentage",
  "definition": "http://unitsofmeasure.org/ucum.html#para-29"
},
"constraint": {
   "type": "AllowedValues",
   "interval": [0 100],
   "significantFigures": 3
}
}
```

b) taskingParameters in Task

```
{
  "open_percentage": 30.5
}
```

5.6. Time

Time and TimeRange are now focusing the time to display, not for the task scheduling purpose.

Example 7. Set clock time

a) taskingParameters in TaskCapability

```
{
   "type": "Time",
   "name": "time",
   "label": "set clock time",
   "description": "The time that set on clock"
}
```

b) taskingParameters in Task

```
{
   "time": "2018-05-02T23:45:32Z"
}
```

5.7. Location

Example 8. Set drone flight destination

a) taskingParameters in TaskCapability

```
{
   "type": "Location",
   "name": "destination",
   "label": "drone landing location",
   "description": "The location that set to drone to land"
}
```

b) taskingParameters in Task

```
{
   "destination”: {
     "type": "Point",
     "coordinates": [-114.06,51.05]
    }
}
```

5.8. CategoryRange

Configure (1) the condition for something to happen e.g., a light changes color based on the level of hurricanes; and (2) the behavior of light can be configured by the tasking. e.g., in country A, red might mean level 4~5, but in country B, red might mean level 3~5. This can be used when we deploy the light in different countries for their conventions.

Example 9. Set light color for a hurricane

a) taskingParameters in TaskCapability

```
{
   "type": "CategoryRange",
   "name": " red_levels",
   "label": "turn on red light when the hurricane is in this range",
   "description": "Set light color to red when the hurricane is in the range",
   "constraint": {
   "type": "AllowedTokens",
   "value": ["level 1","level 2","level 3","level 4","level 5","level 6"]
 }
}
```

b) taskingParameters in Task

```
{
   "red_levels": ["level 4" "level 6"]
}
```

5.9. CountRange

CountRange is similar to CategoryRange in that it configures the condition for something to happen and the behavior of the device being tasking.

Set light color or a queue display based on the count of a queue.

Example 10. Set elevator warning sound based on the number of people

a) taskingParameters in TaskCapability

```
{
  "type": "CountRange",
  "name": "set_elevator_warning",
  "label": "set elevator warning sound based on the number of people",
  "description": "Set the elevator warning sound based on the number of people enter
the elevator",
  "constraint": {
    "type": "AllowedValues",
    "interval": [1 10]
  }
}
```

b) taskingParameters in Task

```
{
  "set_elevator_warning": [6 9]
}
```

5.10. QuantityRange

QuantityRange is similar to CategoryRange in that it configures the condition for something to happen and the behavior of the device being tasking.

Example 11. Set light color based on the indoor air quality

a) taskingParameters in TaskCapability

```
{
  "type": "QuantityRange",
  "name": "good_air_quality_range",
  "label": "Set light color to green if the air quality is good",
  "description": "Set light color to green if the air quality is good",
  "unitOfMeasurement": {
    "symbol": "ug/m3",
    "name": "microgram per cubic meter",
    "definition": " http://www.ess.co.at/AIR-EIA/units.html"
  },
  "constraint": {
    "type": "AllowedValues",
    "interval": [0 500],
    "significantFigures": 4
  }
}
```

b) taskingParameters in Task

```
{
  "good_air_quality_light_color": [0.0 12.0]
}
```

5.11. TimeRange

Example 12. Set calendar to display time range

a) taskingParameters in TaskCapability

```
{
  "type": "TimeRange",
  "name": "calendar_timerange",
  "label": "Calendar display time range",
  "description": "Set the time range to calendar"
}
```

b) taskingParameters in Task

```
{
  "calendar_range": ["2018-05-01T00:00:00Z" "2018-05-31T24:00:00Z"]
}
```

5.12. Union of simple types

The union are the combination of different simple types mentioned above.

Example 13. Control the washing machine

a) taskingParameters in TaskCapability

```
{
  "type": "DataRecord",
  "field": [{
      "type": "Boolean",
      "name": "status",
      "label": "Washing machine on/off status",
      "description": "If set true, the wash machine is on; Otherwise, the washing
machine is off"
    },
    {
      "type": "Category",
      "name": "program",
      "label": "Washing program",
      "description": "The washing program to use",
      "constraint": {
        "type": "AllowedTokens",
        "value": ["Cotton", "Synthetics", "Wool"]
      }
    },
    {
      "type": "Quantity",
      "name": "temperature",
      "label": "Washing temperature",
      "description": "The washing temperature to wash on",
      "unitOfMeasurement": {
        "symbol": "°C",
        "name": "degree Celsius",
        "definition": "http://unitsofmeasure.org/ucum.html#para-30"
      },
      "constraint": {
        "type": "AllowedValues",
        "value": [30, 40, 60]
      }
    },
    {
      "type": "Boolean",
      "name": "quick",
      "label": "Speed program",
      "description": "If set true, the washing time is halved"
    }
  ]
}
```


b) taskingParameters in Task

```
{
  "status": true,
  "program": "Cotton",
  "temperature": 30,
  "quick": true
}
```

Example 14. Task a harvester regarding the areas to harvest

```
a)  taskingParameters in TaskCapability
{
  "type": "DataRecord",
  "field": [{
      "type": "Category",
      "name": "mode",
      "label": "Working mode",
      "description": "The harvesting program to use",
      "constraint": {
        "type": "AllowedTokens",
        "value": ["auto", "wheat", "canola", "corn"]
      }
    },
    {
      "type": "Location",
      "name": "harvest_area",
      "label": "Working area",
      "description": "Set working area to the harvester"
    },
    {
      "type": "Quantity",
      "name": "fan_speed",
      "label": "working fan speed",
      "description": "Set fan speed to the harvester",
      "unitOfMeasurement": {
        "symbol": "rpm",
        "name": "rotational speed",
        "definition": " https://en.wikipedia.org/wiki/Rotational_speed"
      },
      "constraint": {
        "type": "AllowedValues",
        "Interval": [1050 1500],
        "significantFigures": 4
      }
    }
  ]
}
```

b) taskingParameters in Task

```
{
  "mode": "auto",
  "clean_area": {
    "type": "Polygon",
    "coordinates": [
      [100.0, 0.0],
      [110.0, 0.0],
      [110.0, 10.0],
      [100.0, 10.0],
      [100.0, 0.0]
    ]
  },
  "fan_speed": 1200
}
```

Example 15. Control the light

a) taskingParameters in TaskCapability

```
{
  "type": "DataRecord",
  "field": [{
      "type": "Boolean",
      "name": "status",
      "label": "On/Off status",
      "description": "If true, the light is on; Otherwise, the light is off"
    },
    {
      "type": "Text",
      "name": "color",
      "label": "Light color",
      "description": "Specifies the light color in HEX format.",
      "constraint": {
        "type": "AllowedTokens",
        "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
      }
    },
    {
      "type": "Quantity",
      "name": "brightness",
      "label": "Light brightness",
      "description": "Set the light brightness in percentage",
      "unitOfMeasurement": {
        "symbol": "%",
        "name": "percentage",
        "definition": "http://unitsofmeasure.org/ucum.html#para-29"
      },
      "constraint": {
        "type": "AllowedValues",
        "interval": [0 100],
        "significantFigures": 2
      }
    }
  ]
}
```

b) taskingParameters in Task

```
{
     "On/Off": true,
     "color": "#0080FF",
     "brightness": 80
}
```

Example 16. Configure the light color based on air quality readings range

a) taskingParameters in TaskCapability

```
{
  "type": "DataRecord",
  "field": [{
      "type": "QuantityRange",
      "name": "good_air_quality_range",
      "label": "Set light color to green if the air quality is good",
      "description": "Set light color to green if the air quality is good",
      "unitOfMeasurement": {
        "symbol": "ug/m3",
        "name": "microgram per cubic meter",
        "definition": " http://www.ess.co.at/AIR-EIA/units.html"
      },
      "constraint": {
        "type": "AllowedValues",
        "interval": [0 500],
        "significantFigures": 4
      }
    },
    {
      "type": "QuantityRange",
      "name": "medium_air_quality_range",
      "label": "Set light color to yellow if the air quality is medium",
      "description": "Set light color to yellow if the air quality is medium",
      "unitOfMeasurement": {
        "symbol": "ug/m3",
        "name": "microgram per cubic meter",
        "definition": " http://www.ess.co.at/AIR-EIA/units.html"
      },
      "constraint": {
        "type": "AllowedValues",
        "interval": [0 500],
        "significantFigures": 4
      }
    },
    {
      "type": "QuantityRange",
      "name": "bad_air_quality_range",
      "label": "Set light color to red if the air quality is bad",
      "description": "Set light color to red if the air quality is bad",
      "unitOfMeasurement": {
        "symbol": "ug/m3",
        "name": "microgram per cubic meter",
        "definition": " http://www.ess.co.at/AIR-EIA/units.html"
      },
      "constraint": {
        "type": "AllowedValues",
        "interval": [0 500],
        "significantFigures": 4
      }
    }
  ]
```

```
}
```

b) taskingParameters in Task

```
{
  "good_air_quality_light_color": [0 12.0],
  "medium_air_quality_light_color": [12.1 55.4],
  "bad_air_quality_light_color": [55.5 500]
}
```

Appendix A: Revision History

Table 2. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
June 15, 2016	Steve Liang	.1	all	First version