Open Geospatial Consortium

Submission Date: 2017-10-13

Approval Date: 2018-06-01

Publication Date: 2019-01-08

External identifier of this OGC® document: http://www.opengis.net/doc/IS/sensorthings-part2-TaskingCore/1.0

Internal reference number of this OGC® document: 17-079r1

Version: 1.0

Category: OGC® Implementation Standard

Editor: Steve Liang, Tania Khalafbeigi

OGC SensorThings API Part 2 – Tasking Core

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OGC® Standard
Document subtype: Implementation
Document stage: Approved
Document language: English

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.

Contents

i. Abstract

The OGC SensorThings API [OGC 15-078r6] provides an open, geospatial-enabled and unified way to interconnect the Internet of Things (IoT) devices, data, and applications over the Web. At a high level, the OGC SensorThings API provides two main functions and each function is handled by the Sensing part or the Tasking part. The Sensing part provides a standard way to manage and retrieve observations and metadata from heterogeneous IoT sensor systems. The Tasking part provides a standard way for parameterizing - also called tasking - of taskable IoT devices, such as individual sensors and actuators, composite consumer / commercial / industrial / smart cities *in-situ* platforms, mobile and wearable devices, or even unmanned systems platforms such as drones, satellites, connected and autonomous vehicles, etc. This document specifies core of the SensorThings Tasking part.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, ogc documents, iot, internet of things, sensor things, sensors, swe, sensor webs, sensor web enablement, sensor networks

iii. Preface

The OGC SensorThings API is an open standard. This means that anyone can implement the standard on a non-discriminatory, royalty-free basis. The standard is also platform independent. Although SensorThings is a new standard, the requirements build on a rich set of proven-working and widely-adopted open standards, such as Web protocols and the OGC Sensor Web Enablement (SWE) standards, including the ISO/OGC Observation and Measurement data model [OGC 10-004r3 and ISO 19156:2011]. As a result, the OGC SensorThings API is extensible and can be applied to not only simple but also complex use cases.

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.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- SensorUp Inc., Canada
- University of Calgary, Canada

- Fraunhofer-Gesellschaft, Germany
- CGI Federal, USA

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Affiliation
Steve Liang	University of Calgary / SensorUp Inc.
Tania Khalafbeigi	University of Calgary / SensorUp Inc.
Kan Luo	University of Calgary
Brian Miles	CGI Federal
Hylke van der Schaaf	Fraunhofer-Gesellschaft
Michael Jacoby	Fraunhofer-Gesellschaft
Marcus Alzona	Keys

1. Scope

The OGC SensorThings API provides an open standard-based and geospatial-enabled framework to interconnect the Internet of Things (IoT) devices, data, and applications over the Web. The Tasking part provides a standard way for parameterizing - also called tasking - of taskable IoT devices, such as individual sensors and actuators, composite consumer / commercial / industrial / smart cities *in-situ* platforms, mobile and wearable devices, or even unmanned systems platforms such as drones, satellites, connected and autonomous vehicles, etc. This document is the core of the SensorThings Tasking part. This document follows the CRUD functions, semantics and also query options from OGC SensorThings API part 1: Sensing.

2. Conformance

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

The following table lists the requirements classes defined in this standard.

NOTE: The content of Requirements column in the following table is the path fragment that appended to the following URI: http://www.opengis.net/spec/iot_tasking/1.0/, and it provides the URI that can be used to unambiguously identify the requirement and the conformance class.

Requirements class id	Requirements	Description
req/tasking-capability	• req/tasking-capability/properties • req/tasking-capability/relations	`TaskingCapability` entity
req/task	• req/task/properties • req/task/relations	`Task` entity
req/actuator	• req/actuator/properties • req/actuator/relations	`Actuator` entity
req/create-tasks	• req/create-tasks/task-creation	creating `Task` entity
req/create-tasks-via-mqtt	• req/create-tasks-via-mqtt/task-creation	creating `Tasks` through MQTT
req/receive-updates-via-mqtt	• req/receive-updates-via-mqtt/receive-updated	Receiving `Tasks` updates through MQTT

3. 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: OGC 08-094r1, OGC SWE Common Data Model Encoding Standard, 2011

OGC: OGC 15-078r6, OGC SensorThings API — Part 1: Sensing, 2016

OGC: OGC 17-011r2, JSON Encoding Rules SWE Common / SensorML, 2018

OASIS: OASIS MQTT Version 3.1.1, 2014

ISO: ISO 8601:2004 Data elements and interchange formats – Information interchange - Representation of dates and times, 2004

OGC: OGC 10-004r3 and ISO 19156:2011(E), OGC Abstract Specification Topic 20: Geographic information — Observations and Measurements, 2010

OASIS: OASIS OData Version 4.0 Part 1: Protocol Plus Errata 02, 2014

OASIS: OASIS OData Version 4.0 Part 2: URL Conventions Plus Errata 02, 2014

OASIS: OASIS OData JSON Format Version 4.0 Plus Errata 02, 2014

OASIS: OASIS OData ABNF Construction Rules Errata 02, 2014

IETF: RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, 1999

OGC: OGC 12-000, OGC® SensorML: Model and XML Encoding Standard, 2014

4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r9], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

For the purposes of this document, the following additional terms and definitions apply.

4.1
Actuator
A type of transducer that converts a signal to some real-world action or phenomenon. [OGC 12-000]

4.2
Collection
Sets of Resources, which can be retrieved in whole or in part. [RFC5023]

4.3
Entity
Entities are instances of entity types. [OASIS OData Version 4.0 Part 1: Protocol Plus Errata 02]

Note: Thing, Actuator, Task are some example entity types of the OGC SensorThings API.

4.4
Entity sets
Entity sets are named collections of entities (e.g. Tasks is an entity set containing Task entities). An entity's key uniquely identifies the entity within an entity set. Entity sets provide entry points into an OGC SensorThings API service. [OASIS OData Version 4.0 Part 1: Protocol Plus Errata 02]

4.5
(Internet of) Thing
A thing is an object of the physical world (physical things) or the information world (virtual things) that is capable of being identified and integrated into communication networks. [ITU-T Y.2060]

4.6
Measurement
A set of operations having the object of determining the value of a quantity [OGC 10-004r3 / ISO 19156:2011]

4.7
Observation
Act of measuring or otherwise determining the value of a property [OGC 10-004r3 / ISO 19156:2011]

4.8
Observation Result
Estimate of the value of a property determined through a known observation procedure [OGC 10-004r3 / ISO 19156:2011]

4.9
Resource
A network-accessible data object or service identified by an URI, as defined in [RFC 2616]

4.10
REST
The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system. REST focuses on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. It encompasses the fundamental constraints upon components, connectors, and data that define the basis of the Web architecture, and thus the essence of its behavior as a network-based application. An API that conforms to the REST architectural principles/constraints is called a RESTful API. [OGC 15-078r6]

4.11
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]

5. Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1 Identifiers
The normative provisions in this specification are denoted by the URI

http://www.opengis.net/spec/iot_tasking/1.0/

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

6. Symbols and abbreviated terms

API	Application Programming Interface
CAV	Connected and Autonomous Vehicles
CRUD	Create, Read, Update, and Delete
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
OData	the Open Data Protocol
OGC	Open Geospatial Consortium
REST	REpresentational State Transfer
SensorML	Sensor Model Language
SWE	Sensor Web Enablement
UML	Unified Modeling Language
UxS	Unmanned Systems (aerial, underwater, and other unmanned vehicle platforms)
XML	eXtensible Markup Language

7. The SensorThings API Tasking Entities

`TaskingCapability`, `Actuator`, and `Task` are the main entities for Tasking core. These entities SHALL be listed in the response to a GET request to the root URL as described Sensing part. In case the server implementation only has Tasking part, the index SHALL contain `Things`, `TaskingCapabilities`, `Tasks`, and `Actuators`.

7.1 The Tasking Entities

The SensorThings API Tasking core's Entities are depicted in Figure 1.

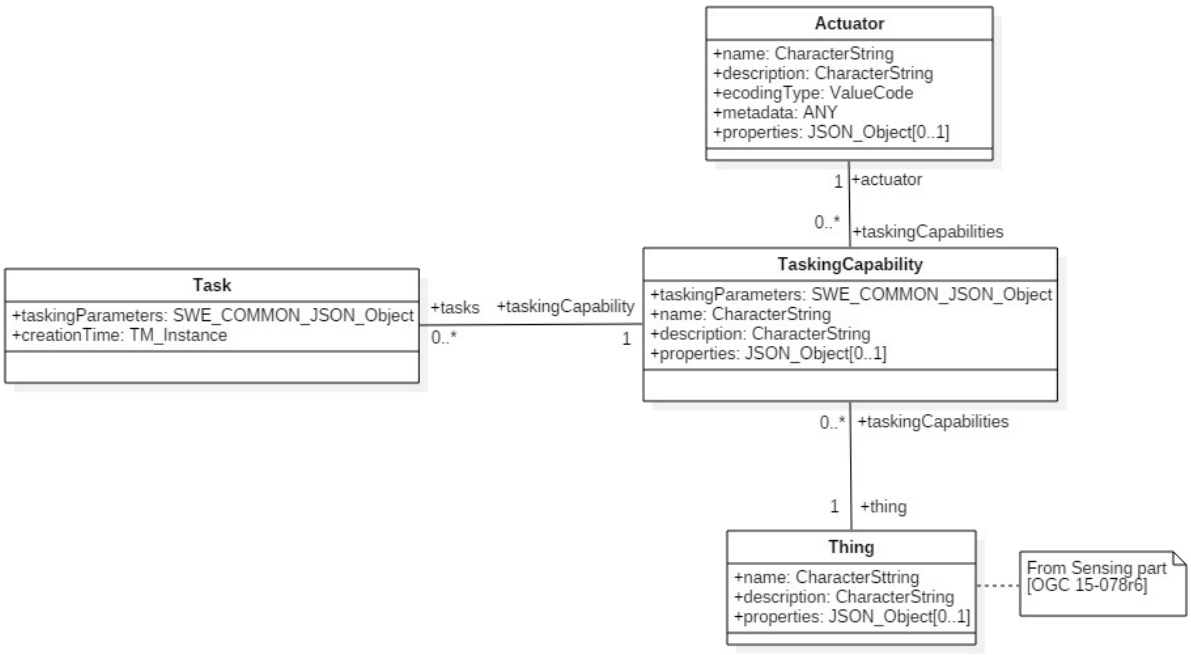

Figure 1 Tasking Entities

In this section, the properties for each entity types and the direct relation to the other entity types are explained. In addition, for each entity type we show an example of the associated JSON encoding.

7.1.1 Thing

This entity is from Sensing part. We add the information here to make this documentation complete in case a server is only implementing the Tasking core. Please note that if the server is implementing both Sensing and Tasking parts, the direct relationship for Thing entity is the combination of the relationship defined in this document with the relationships defined in the Sensing part.

The OGC SensorThings API follows the ITU-T definition, *i.e.*, with regard to the Internet of Things, a thing is an object of the physical world (physical things) or the information world (virtual things) that is capable of being identified and integrated into communication networks [ITU-T Y.2060].

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/thing	
Target type	Web Service
Dependency	**http://www.opengis.net/spec/iot_sensing/1.0/req/thing**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/thing/properties** *Each* `Thing` *entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 1.*
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/thing/relations**

	Each `Thing` entity SHALL have the direct relation between a `Thing` entity and other entity types listed in Table 2.

Table 1 Properties of a `Thing` entity

Name	Definition	Data type	Multiplicity and use
`name`	A property provides a label for `Thing` entity, commonly a descriptive name.	CharacterString	One (mandatory)
`description`	This is a short description of the corresponding `Thing` entity.	CharacterString	One (mandatory)
`properties`	A JSON Object containing user-annotated properties as key-value pairs.	JSON Object	Zero-to-one

Table 2 Direct relation between a `Thing` entity and other entity types

Entity type	Relation	Description
`TaskingCapabilities`	One mandatory to many optional	A `Thing` MAY have zero-to-many `TaskingCapabilities`.

Example 1 an example of a `Thing` entity implementing the Tasking core

```
{
  "@iot.id": 1,
  "@iot.selfLink": "http://example.org/v1.0/Things(1)",
  "TaskingCapabilities@iot.navigationLink": "Things(1)/TaskingCapabilities",
  "name": "LinkitSmart with LED",
  "description": "LinkitSmart device with the LED that can be turned on/off
and the color can be changed."
}
```

7.1.2 TaskingCapability

The `TaskingCapability` entity contains information about the capabilities of the taskable device. It contains all the parameters that can be used for controlling the device. SWE Common JSON encoding rules [OGC17-011r2] are used to define these parameters for `TaskingCapability`.

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/tasking-capability	
Target type	Web Service
Dependency	**http://www.opengis.net/spec/iot_sensing/1.0/req/entity-control-information/common-control-information**
Dependency	**http://www.opengis.net/doc/BP/SWE-JSON/1.0/req**
Dependency	**http://www.opengis.net/doc/IS/SWE/2.0**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/tasking-capability/properties**

	Each `TaskingCapability` *entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 3.*
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/tasking-capability/relations** *Each* `TaskingCapability` *entity SHALL have the direct relation between a* `TaskingCapability` *entity and other entity types listed in Table 4.*

Table 3 Properties of a `TaskingCapability`

Name	Definition	Data type	Multiplicity and use
`name`	A property provides a label for the entity, commonly a descriptive name.	CharacterString	One (mandatory)
`description`	This is a short description of the corresponding entity.	CharacterString	One (mandatory)
`taskingParameters`	The `taskingParameters` property describes optional and mandatory tasking parameters. Clients use the definition to provide corresponding tasking parameter values. To ensure common understanding between client and server, a common 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`.	SWE Common JSON Object	One (mandatory)
`properties`	A JSON Object containing user-annotated properties as key-value pairs.	JSON Object	Zero-to-one

Table 4 Direct relation between a `TaskCapability` and other entity types

Entity Type	Relation	Description
Task	One mandatory to many optional	A `TaskingCapability` has zero-to-many `Tasks`. A `Task` has one-and-only-one `TaskingCapability`.
Thing	Many optional to one mandatory	A `TaskingCapability` has one-and-only-one `Thing`. A `Thing` has zero-to-many `TaskingCapabilities`.
Actuator	Many optional to one mandatory	A `TaskingCapability` has one-and-only-one `Actuator`. An `Actuator` has zero-to-many `TaskingCapabilities`.

Example 2 an example of a `TaskCapability` entity

```
{
  "@iot.id": 1,
  "@iot.selfLink": "http://example.org/v1.0/TaskingCapabilities(1)",
```

```
  "Thing@iot.navigationLink": "TaskingCapabilities(1)/Thing",

  "Actuator@iot.navigationLink": "TaskingCapabilities(1)/Actuator",

  "Tasks@iot.navigationLink": "TaskingCapabilities(1)/Tasks",

  "name": "Control Light",

  "description": "Turn the light on and off, as well as specifying light
color. ",

  "taskingParameters": {

    "type": "DataRecord",

    "field": [

      {

        "name": "status",

        "label": "On/Off status",

        "description": "Specifies turning the light On or Off",

        "type": "Category",

        "constraint": {

          "type": "AllowedTokens",

          "value": [

            "on","off"

          ]

        }

      },

      {

        "name": "color",

        "label": "Light Color",

        "description": "Specifies the light color in RGB HEX format.
Example: #FF11A0",

        "type": "Text",

        "constraint": {

          "type": "AllowedTokens",
```

```
          "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
        }
      }
    ]
  }
}
```

7.1.3 Task

`Task` entity contains the parameter detail of the control action that should be run on the task-able device.

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/task	
Target type	Web Service
Dependency	**http://www.opengis.net/spec/iot_sensing/1.0/req/entity-control-information/common-control-information**
Dependency	**http://www.opengis.net/doc/BP/SWE-JSON/1.0/req**
Dependency	**http://www.opengis.net/doc/IS/SWE/2.0**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/task/properties** *Each* `Task` *entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 5.*
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/task /relations** *Each* `Task` *entity SHALL have the direct relation between a* `Task` *entity and other entity types listed in Table 6*

Table 5 Properties of a `Task`

Name	**Definition**	**Data type**	**Multiplicity and use**
`taskingParameters`	The `taskingParameters` property describes values for optional and mandatory tasking parameters. Clients use the definition to provide corresponding tasking parameter values. To ensure common understanding between client and server, a common exchange protocol is used to express both descriptions and tasking parameter values. SensorThings uses the JSON encoding for SWE Common data block defined in OGC 08-094r1 to define `taskingParameters`. `taskingParameters` is a SWE Common data block and MUST have key-value pairs in a JSON object. Key MUST be the name	SWE Common JSON Object	One (mandatory)

	described in `TaskingCapablity`'s `taskingParamaters` and value MUST be the value of that parameter for this `Task`.		
`creationTime`	The time when the `Task` is created. This time SHALL only be added automatically by the service.	TM_Instant (ISO-8601 Time String)	One (mandatory)

Table 6 Direct relation between a `Task` and other entity types

Entity Type	Relation	Description
`TaskingCapability`	Many optional to one mandatory	A `Task` has one-and-only-one `TaskingCapability`. A `TaskingCapability` has zero-to-many `Tasks`.

Example 3 an example of a `Task` entity

```
{
  "@iot.id": 2,
  "@iot.selfLink": "http://example.org/v1.0/Tasks(1)",
  "TaskingCapability@iot.navigationLink": "Tasks(2)/TaskingCapability",
  "creationTime": "2017-01-01T00:00:00.000Z",
  "taskingParameters": {
    "status": "on",
    "color": "#FF0000"
  }
}
```

7.1.4 Actuator

An `Actuator` is a device that can be controlled/tasked. The `Actuator` entity contains information and metadata about taskable actuator. Each `TaskingCapability` has one `Actuator` and defines the parameters that can be set/tasked for the `Actuator`.

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/actuator	
Target type	Web Service
Dependency	**http://www.opengis.net/spec/iot_sensing/1.0/req/entity-control-information/common-control-information**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/actuator/properties** *Each* `Actuator` *entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 7.*
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/actuator/relations**

	Each **`Actuator`** *entity SHALL have the direct relation between a* `Task` *entity and other entity types listed in Table 8 .*

Table 7 Properties of an `Actuator`

Name	Definition	Data type	Multiplicity and use
`name`	A property provides a label for the entity, commonly a descriptive name.	CharacterString	One (mandatory)
`description`	The description of the Actuator entity	CharacterString	One (mandatory)
`encodingType`	The encoding type of the `metadata` property. Its value is one of the ValueCode enumeration (see for Table 9 for the available ValueCode).	Any (depending on the value of the `encodingType`)	One (mandatory)
`metadata`	The detailed description of the `Actuator`. The `metadata` type is defined by `encodingType`.	Any (depending on the value of the `encodingType`)	One (mandatory)
`properties`	A JSON Object containing user-annotated properties as key-value pairs.	JSON Object	Zero-to-one

Table 8 Direct relation between an `Actuator` and other entity types

Entity Type	Relation	Description
`TaskingCapabilities`	One mandatory to many optional	An `Actuator` has zero-to-many `TaskingCapabilities`. A `TaskingCapability` has one-and-only-one `Actuator`.

Table 9 List of some code values used for identifying types for the `encodingType` of the `Actuator` entity

`Actuator encodingType`	**ValueCode Value**
PDF	`application/pdf`
SensorML	`http://www.opengis.net/doc/IS/SensorML/2.0`

The `Actuator encodingType` allows clients to know how to interpret `metadata`'s value(s). Currently, the SensorThings API defines two common `Actuator metadata encodingTypes`. Most sensor manufacturers provide their sensor datasheets in a PDF format. As a result, PDF is a `Sensor encodingType` supported by SensorThings API. The second `Sensor encodingType` is SensorML.

Example 4 an example of an `Actuator` entity

```
{
  "@iot.id": 3,
  "@iot.selfLink": "http://example.org/v1.0/Actuators(3)",
```

```
  "TaskingCapabilities@iot.navigationLink":
"Actuators(3)/TaskingCapabilities",

  "name": "Linkit Smart 7688 Duo Board with LED",

  "description": "Linkit Smart 7688 Duo Board that has an LED which can
be tasked as on/off with different color.",

  "encodingType": "application/pdf",

  "metadata": "http://example.org/linkit_7688.pdf"

}
```

8. SensorThings Tasking Create

The SensorThings API Tasking part SHALL support creating `Task` entities. The server checks the `taskingParamaters` to be compliant to the corresponding `TaskingCapability`. If the `taskingParameters` are correct, the `Task` will be created.

This specification contains only the core functionalities for Tasking part. All other entities than `Task`, including `TaskingCapabilities`, `Actuators`, and `Things` are pre-provisioned to the Server and the client can only create `Tasks`.

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/create-tasks	
Target type	Web Service
Dependency	**http://docs.oasis-open.org/odata/odata/v4.0/errata02/os/complete/part1-protocol/odata-v4.0-errata02-os-part1-protocol-complete.html#_Toc406398326**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/create-tasks/task-creation** *To create a* `Task` *in a collection, the client SHALL send a HTTP* `POST` *request to* `Task` *collection's URL. The* `POST` *body SHALL contain a single valid* `Task` *representation.* *When clients create* `Tasks` *in a SensorThings service, they SHALL follow the integrity constraints listed in Table 10.* *When a SensorThings service receives a create* `Task` *request, the service SHALL set the* `createdTime` *property of the entity to current server time.* *SensorThings API services SHALL support linking new* `Task` *entities to existing entities upon creation. To create a new* `Task` *with links to existing entities, the client SHALL include the unique identifiers of the related entities associated with the corresponding navigation properties in the request body.*

Table 10 Integrity constraints when creating a `Task` entity

Scenario	Integrity Constraints
Create a `Task` entity	SHALL link to a `TaskingCapability` entity

8.1 Link to existing entities when creating a `Task` entity

Each `Task` has one `TaksingCapability`. There are two approaches to create a `Task`. (1) The corresponding `TaskingCapability` is specified in the request with its @iot.id; and (2) The corresponding `TaskingCapability` is specified in the POST URL, *i.e.*, `/v1.0/TaskingCapabilities({id})/Tasks`.

HTTP Method: `POST`

URI Pattern: `SERVICE_ROOT_URI/Tasks`

Header: `Content-Type: application/json`

Message Body: A single valid `Task` entity representation.

Example 5 create a `Task` entity, which links to an existing `TaskingCapability` entity (whose id is 1).

```
POST /v1.0/Tasks HTTP/1.1

Host: example.org/
Content-Type: application/json

{
    "taskingParameters": {

      "status": "on",

      "color": "#FF0000"

    },

    "TaskingCapability": {
      "@iot.id": 1
    }
}
```

HTTP Method: `POST`

URI Pattern: `SERVICE_ROOT_URI/TaskingCapabilities({id})/Tasks`

Header: `Content-Type: application/json`

Message Body: A single valid `Task` entity representation.

Example 6 create a `Task` entity, which links to an existing `TaskingCapability` entity (whose id is 1) by specifying `TaskingCapability` id in URL.

```
POST /v1.0/TaskingCapabilities(1)/Tasks HTTP/1.1

Host: example.org/
Content-Type: application/json

{
  "taskingParameters": {

    "status": "on",

    "color": "#FF0000"

  }

}
```

9. SensorThings Tasking MQTT Extension

In addition to support HTTP protocol, a SensorThings service MAY support the Message Queuing Telemetry Transport (MQTT) protocol[1] to enhance the SensorThings service publish and subscribe capabilities. This section describes the SensorThings MQTT extension.

9.1 Create a SensorThings `Task` with MQTT Publish

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/create-tasks-via-mqtt	
Target type	Web Service
Dependency	**http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/create-tasks-via-mqtt/tasks-creation** *In order to allow clients to create* `Tasks` *with MQTT Publish, a service SHALL support the creation of* `Tasks` *with MQTT as defined in Section 9.1.*

The SensorThings MQTT extension provides the capability of creating `Task` entity using MQTT protocol. To create a `Task` entity in MQTT, the client sends a `MQTT Publish` request to the SensorThings service and the MQTT topic is the `Tasks` resource path. The MQTT application message contains a single valid `Task` entity representation. Figure 2 contains the sequence diagram for creating `Task` using MQTT publish as well as MQTT sending notifications for `Task` creation.

[1] MQTT version 3.1.1 is an OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

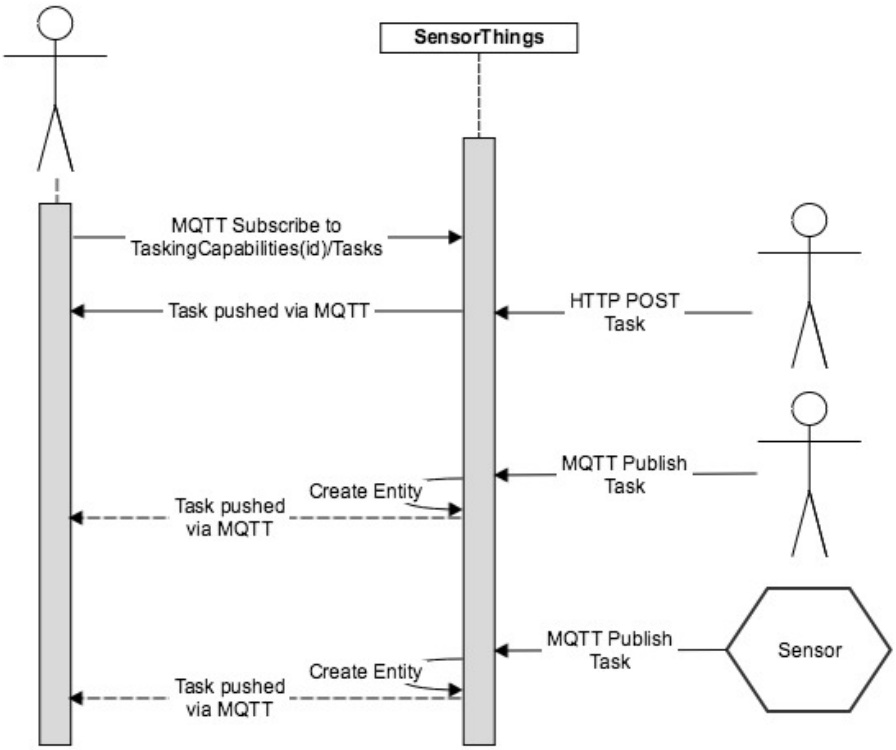

Figure 2 Creating `Tasks` using MQTT publish, and receive notifications for `Tasks` with MQTT

If the MQTT topic for the `Task` is a `navigationLink` from `TaskingCapability`, the new `Task` entity is automatically linked to that `TaskingCapability` respectively.

Similar to creating `Tasks` with `HTTP POST`, creating `Tasks` with `MQTT Publish` follows the integrity constraints for creating `Task` as listed in Table 10.

9.1.1 Link to existing entities when creating a `Task` entity

To link to existing entities when creating a `Task` entity with MQTT, the conditions specified in Section 8 are applied.

9.2 Receive updates with MQTT Subscribe

Requirements Class	
http://www.opengis.net/spec/iot_tasking/1.0/req/receive-updates-via-mqtt	
Target type	Web Service
Dependency	**http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html**
Requirement	**http://www.opengis.net/spec/iot_tasking/1.0/req/receive-updates-via-mqtt/receive-updates** *To allow clients to receive notifications for the updates of SensorThings tasking entities with MQTT, a service SHALL support the receiving updates with MQTT Subscribe as defined in Section 9.2.*

To receive notifications from a SensorThings service when some tasking entities are updated, a client can send a MQTT Subscribe request to the SensorThings service. The SensorThings API defines the following MQTT subscription use cases.

Receiving notifications from a SensorThings service follows the requirement **http://www.opengis.net/spec/iot_sensing/1.0/req/receive-updates-via-mqtt** of the Sensing part.

When the SensorThings MQTT extension is being used for controlling an Actuator, the actuator (gateway) can subscribe to `Tasks` and whenever it receives a `Task` over MQTT, it can perform it. Figure 3 shows a sequence diagram of this process.

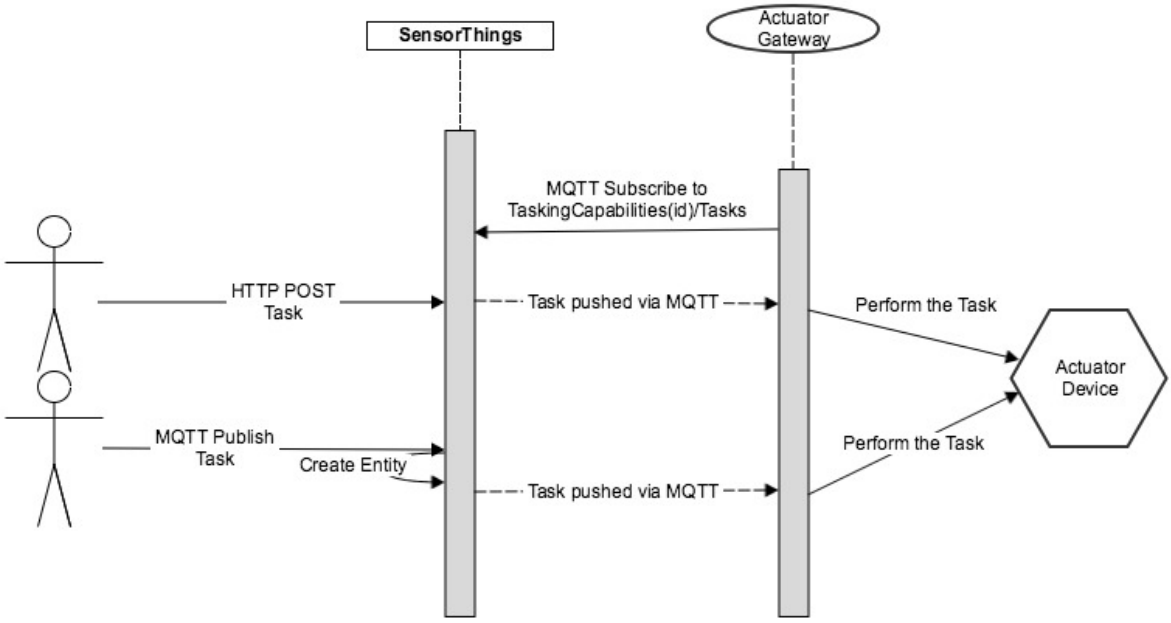

Figure 3 **Actuator communication to SensorThings via MQTT**

Annex A: Conformance Class Abstract Test Suite (Normative)

A.1 SensorThings Tasking Core Tests

A.1.1 Conformance class: SensorThings API `TaskingCapability` Entity

Conformance class id: http://www.opengis.net/spec/iot_tasking/1.0/conf/tasking-capability

Test: `TaskingCapability` Entity

Requirements	• http://www.opengis.net/spec/iot_tasking/1.0/conf/tasking-capability/properties • http://www.opengis.net/spec/iot_tasking/1.0/conf/tasking-capability/relations
Test Purpose	Check if each `TaskingCapabilitiy` entity has the mandatory properties and mandatory relations as defined in this standard.
Test Methods	Inspect the full JSON object of the `TaskingCapability` entity sets (*i.e.*, without `$select`) to identify, if each entity has the mandatory properties defined in the corresponding requirement. Inspect the full JSON object of each `TaskingCapabilities` entity set (*i.e.*, without using the `$select` query option) to identify, if each entity has the mandatory relations (*i.e.*, `@iot.navigationLink`) defined in the corresponding requirement.

A.1.2 Conformance class: SensorThings API `Task` Entity

Conformance class id: http://www.opengis.net/spec/iot_tasking/1.0/conf/task

Test: `Task` Entity

Requirements	• http://www.opengis.net/spec/iot_tasking/1.0/conf/task/properties • http://www.opengis.net/spec/iot_tasking/1.0/conf/task/relations
Test Purpose	Check if each `Task` entity has the mandatory properties and mandatory relations as defined in this standard.
Test Methods	Inspect the full JSON object of the `Tasks` entity sets (*i.e.*, without `$select`) to identify, if each entity has the mandatory properties defined in the corresponding requirement. Inspect the full JSON object of each `Tasks` entity set (*i.e.*, without using the `$select` query option) to identify, if each entity has the mandatory relations (*i.e.*, `@iot.navigationLink`) defined in the corresponding requirement.

A.1.3 Conformance class: SensorThings API `Actuator` Entity

Conformance class id: http://www.opengis.net/spec/iot_tasking/1.0/conf/actuator

Test: `Actuator` Entity

Requirements	• http://www.opengis.net/spec/iot_tasking/1.0/conf/actuator/properties • http://www.opengis.net/spec/iot_tasking/1.0/conf/actuator/relations
Test Purpose	Check if each `Actuator` entity has the mandatory properties and mandatory relations as defined in this standard
Test Methods	Inspect the full JSON object of the `Actuator` entity sets (*i.e.*, without `$select`) to identify, if each entity has the mandatory properties defined in the corresponding requirement. Inspect the full JSON object of each `Actuator` entity set (*i.e.*, without using the `$select` query option) to identify, if each entity has the mandatory relations (*i.e.*, `@iot.navigationLink`) defined in the corresponding requirement.

A.1.4 A.1.4. Conformance class: SensorThings API Create `Task`

Conformance class id: http://www.opengis.net/spec/iot_tasking/1.0/conf/create-entity

Dependencies:

- http://www.opengis.net/spec/iot_tasking/1.0/conf/task
- http://www.opengis.net/spec/iot_sensing/1.0/conf/resource-path

Requirements	• http://www.opengis.net/spec/iot_sensing/1.0/req/create-update-delete/create-entity
Test purpose	Check if the service supports the creation of `Task` as defined in this specification.
Test method	Create a `Task` entity instance by following the integrity constraints and link to the existing related entities with a single request, check if the `Task` entity instance is successfully created and the server responds as defined in this specification. Issue a `Task` entity creation request that does not follow the integrity constraints, check if the service fails the request without creating the `Task` and responds the appropriate HTTP status code.

A.2 SensorThings Tasking MQTT Extension

A.2.1 Test: SensorThings API `Task` Creation via MQTT

Conformance class id: http://www.opengis.net/spec/iot_tasking/1.0/conf/create-tasks-via-mqtt

Dependencies:

- http://www.opengis.net/spec/iot_tasking/1.0/conf/task

Requirements	• http://www.opengis.net/spec/iot_tasking/1.0/req/create-tasks-via-mqtt/task-creation
Test purpose	Check if the service supports the creation of `Task` entity via MQTT as defined in this standard.
Test method	Create a `Task` entity instance containing binding information for navigation properties using `MQTT Publish`, check if the server responds as defined in this standard.

A.2.2 Test: Receiving Updates for Tasks via MQTT

Conformance class id: http://www.opengis.net/spec/iot_tasking/1.0/conf/ receive-updates-via-mqtt

Dependencies:

- http://www.opengis.net/spec/iot_tasking/1.0/conf/task

Requirements	• http://www.opengis.net/spec/iot_tasking/1.0/req/receive-updates-via-mqtt/receive-updates
Test purpose	Check if a client can receive notifications for the creation of Task entity with MQTT.
Test method	Subscribe to Task entity set with MQTT Subscribe. Then create a new Task. Check if a complete JSON representation of the newly created entity through MQTT is received.

Annex B: Revision history

Date	Release	Author	Paragraph modified	Description
2017-11-30	0.1.0	Steve Liang		Initial draft of SensorThings: Tasking
2017-11-30	0.1.1	Steve Liang		Fixed some typos, corrected some inconsistencies between UML and text
2018-01-08	0.1.2	Steve Liang		Fixed some inconsistencies between this draft and SensorThings Sensing part.
2018-04-02	1.0r1	Tania Khalafbeigi		Updated based on public comments.

Annex C: Bibliography

ITU-T Y.2060 Overview of the Internet of Things, 2012. Available Online: https://www.itu.int/rec/T-REC-Y.2060-201206-I

OGC and ISO 19156:2001, OGC and ISO 19156:2011(E), OGC Abstract Specification: Geographic information — Observations and Measurements. Available Online: http://portal.opengeospatial.org/files/?artifact_id=41579

OGC 12-000, OGC® SensorML: Model and XML Encoding Standard. Available Online: http://www.opengeospatial.org/standards/sensorml

RFC 5023, The Atom Publishing Protocol. Available Online: https://www.ietf.org/rfc/rfc5023.txt

RFC 6902, JavaScript Object Notation (JSON) Patch. Available Online: https://www.ietf.org/rfc/rfc6902.txt

OGC 08-094r1, OGC® SWE Common Data Model Encoding Standard. Available Online: http://www.opengeospatial.org/standards/swecommon

OGC 17-011r2, JSON Encoding Rules for SWE Common and SensorML. Available Online: http://docs.opengeospatial.org/bp/17-011r2/17-011r2.html