|  |
| --- |
| CHANGE REQUEST |
| Meeting ID:\* |  |
| Source:\* |  |
| Date:\* |  |
| Reason for Change/s:\* |  |
| CR against: Release\* |  |
| CR against: WI\* | [x]  Active <WI-100> [ ]  MNT maintenance / < Work Item number(optional)>Is this a mirror CR? Yes [ ]  No [x] mirror CR number: [ ]  STE Small Technical Enhancements / < Work Item number (optional)>Only ONE of the above shall be ticked |
| CR against: TS/TR\* |  |
| Clauses \* | 2.1, 2.2, 5 |
| Type of change: \* | [ ]  Editorial change[x]  Bug Fix or Correction[ ]  Change to existing feature or functionality[ ]  New feature or functionalityOnly ONE of the above shall be ticked |
| Other TS/TR(s) impacted | None |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES [x]  NO [ ] This CR may break backwards compatibility with the last approved version of the TS? YES [ ]  NO [ ]  |
| Template Version: January 2019 (do not modify) |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

GUIDELINES for Change Requests:

Provide an informative introduction containing the problem(s) being solved, and a summary list of proposals.

Each CR should contain changes related to only one particular issue/problem.

In case of a correction, and the change apply to previous releases, a separate “mirror CR” should be posted at the same time of this CR

Mirror CR: applies only when the text, including clause numbering are exactly the same.

Companion CR: applies when the change means the same but the baselines differ in some way (e.g. clause number).

Follow the principle of completeness, where all changes related to the issue or problem within a deliverable are simultaneously proposed to be made E.g. A change impacting 5 tables should not only include a proposal to change only 3 tables. Includes any changes to references, definitions, and acronyms in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar to the extent practicable.

Use Change bars for modifications.

The change should include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change. Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the new clause is proposed to be located.

Multiple changes in a single CR shall be clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.

When subsequent changes are made to content of a CR, then the accepted version should not show changes over changes. The accepted version of the CR should only show changes relative to the baseline approved text.

## Introduction

Introduction of OGC / STA

<https://git.onem2m.org/specifications/ts-0041/-/merge_requests/5>

----------------------- Start of change 1 -----------------------

# 4 Conventions

The key words “Shall”, “Shall not”, “May”, “Need not”, “Should”, “Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.1]

5 Introduction to OGC SensorThings API

The SensorThings API (STA) is a standard of the Open Geospatial Consortium (OGC). It provides a framework for communication and exchanging data between sensors and applications. The standard is devided in two parts. SensorThings API Part 1 is dedicated to sensing and was published in 2016 and updated in 2021 [1].

A STA-based architecture works in client/server mode. A sensor device pushes data to the SensorThings Server via HTTP. A SensorThings Server may also support MQTT protocol to support publish and subscribe capabilities. An interested application can subscribe to the MQTT-Broker, in order to get notified about new sensor events.

Figure 5-1 STA message flow

The data in the SensorThings server are organized as according to **Sensing Entities** (see Figure 5.0-1: Sensing Entities data model [1]).

Figure 5-1 STA Sensing Entities Data Model [1]

In the Sensing Entities Data Model events or sensor data are called “observations”. Before a sensor is able to push an observation to the server it needs at least a ‘Thing’ and a ‘Datastream’ entity. This has to be created beforehand. One ‘Thing’ might have different ‘Sensors’, one ‘Location’ or many ‘HistoricalLocations’.

The Sensing Entities data model and the purpose of data within the data model discloses mainly two data characteristics, associated with a ‘thing’:

- Data observations originated by sensors or commands sent to interact with actuators may be seen as IoT data from oneM2M point of view.

While:

- Data embedded in the Sensing Entities Data Model, like “historic locations” should be seen as data for documentation purposes.

----------------------- End of change 1 -----------------------

----------------------- Start of change 2 -----------------------

## 2.1 Normative references

- [1] OGC SensorThings API “Part 1: Sensing Version 1.1” (http://www.opengis.net/doc/is/sensorthings/1.1)

----------------------- End of change 2 -----------------------

----------------------- Start of change 3 -----------------------

## 2.2 Informative references

Clause 2.2 shall only contain informative references which are cited in the document itself.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

* Use the **EX** style, add the letter “i” (for informative) before the number (which shall be in square brackets) and separate this from the title with a tab (you may use sequence fields for automatically numbering references).

----------------------- End of change 3 -----------------------