|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* |  |
| Source:\* |  |
| Date:\* |  |
| Reason for Change/s:\* |  |
| CR against: Release\* |  |
| CR against: WI\* | Active <WI-100>  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number:  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* |  |
| Clauses \* |  |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only 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  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

Adding intro to data mapping

<https://git.onem2m.org/specifications/ts-0041/-/merge_requests/6>

<https://git.onem2m.org/specifications/ts-0041/-/merge_requests/6/diffs?commit_id=794e7d865da12dd3e9e3a6343ddeae5829705b7b>

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

---a/TS-0041-oneM2M-SensorThings\_interworking.md  
+++b/TS-0041-oneM2M-SensorThings\_interworking.md

@@ -73,6 +73,7 @@ References are either specific (identified by date of publication and/or edition

- <a name="\_ref\_1">[1]</a> OGC SensorThings API "Part 1: Sensing Version 1.1" (http://www.opengis.net/doc/is/sensorthings/1.1)

- <a name="\_ref\_2">[2]</a> oneM2M TS-0033 (V3.0.0): "Interworking Framework"

## 2.2 Informative references

@@ -174,8 +175,62 @@ The Sensing Entities data model and the purpose of data within the data model di

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

## 5.1 User defined subdivisions of clause(s) from here onwards

&lt;Text>

# 6 Architecture Model of OGC/STA to oneM2M interworking

## 6.0 Introduction

Figure 6.0-1 shows an architecture approach for an Interworking Proxy Entity (IPE) between oneM2M and the OGC SensorThings API. The IPE is located between a oneM2M CSE and an OGC/SensorThings API (STA)-Server.

The basic interworking enables applications that are connected to an oneM2M-based system to get data from sensors that are connected to an OGC/STA server. Furthermore, an application that is connected to an OGC/STA server will be able to get data from sensors that are connected to an oneM2M-based system.

<figure>

<img src="media/figure6\_0\_\_1.png" alt="arch\_overview">

<figcaption>Figure 6.0-1: IPE architecture overview with data flow</figcaption>

</figure>

## 6.1 OGC/STA-to-oneM2M Data Model Mapping

According to oneM2M TS-0033 <a href="#\_ref\_2">[2]</a> a representation of a non-oneM2M Proximal IoT function/device in a oneM2M-specified resource instance is to be synchronized with the entity that it represents.

This means that the OGC/STA data model is represented in the hosting CSE. The data in the OGC/STA server are organized as Sensing Entities <a href="#\_ref\_1">[1]</a> (see Figure 5-2: STA Sensing Entities data model).

The oneM2M structure for data models is a tree-structure where data are organized in containers or trees of containers.

The OGC/STA data model is a relational one, as used in databases, and not hierarchical. Thus, it creates a challenge for full interworking of all data captured in the OGC/STA data model. In this technical specification, only a limited set of data is mapped between OGC/STA and oneM2M.

<figure>

<img src="media/ogc\_non\_model.png" alt="non\_mapping">

<figcaption>Figure 6.1-1: OGC data model cannot directly be mapped to oneM2M</figcaption>

</figure>

The SensorThings data model is comprehensive and may be regarded as a n:m relational database structure, holding both:

- sensor (IoT-data); and

- administrative data (like historic locations or historic products IDs).

## 6.2 Architecture Approach

In order to transfer data from a oneM2M sensor to OGC/STA the IPE creates a <subscription> to the <container> resource with the desired data and when a new <contentInstance> is added it gets a <notification> message containing the <contentInstance> resource.

Figure 6.2-1 shows the oneM2M-to-OGC/STA direction. Based upon the creation of the <contentInstance> in the hosting CSE, the IPE gets a <notification> message including the <contentInstance>. The IPE constructs an "Observation" creation request and copies the 'content' attribute of the <contentInstance> to the 'result' attribute of the "Observation" shown in Figure 6.2-2 and sends it to the OGC/STA server.

<figure>

<img src="media/onem2m\_to\_ogc\_flow.png" alt="onem2m\_to\_ogc\_flow">

<figcaption>Figure 6.2-1: IPE oneM2M-to-OGC/STA direction</figcaption>

</figure>

<figure>

<img src="media/content\_copy.png" alt="content\_to\_observation\_copy">

<figcaption>Figure 6.2-2: Copying content from CIN to observation</figcaption>

</figure>

Figure 6.2-3 shows the OGC/STA-to-oneM2M direction. OGC/STA does not provide a publish/subscribe mechanism on HTTP protocol level, but OGC allows an optional MQTT extension for STA services <a href="#\_ref\_1">[1]</a>. The IPE subscribes to the MQTT-Broker of the OGC/STA server. The OGC/STA server publishes its new "Observation" via the MQTT broker. The IPE creates a <contentInstance> using a HTTP request and copies the 'result' attribute of the "Observation" to the 'content' attribute of the <contentInstance>. The <container> should be created beforehand at the hosting CSE where the IPE <contentInstance> resources are stored. All interested applications may subscribe to this <container> resource.

<figure>

<img src="media/ogc\_to\_onem2m\_flow.png" alt="ogc\_to\_onem2m\_flow">

<figcaption>Figure 6.2-3: IPE OGC/STA-to-oneM2M direction</figcaption>

</figure>

This approach is simple and sufficient in cases that only require translating "Observation" to <contentInstances> and vice-versa. Data are stored in the hosting CSE, but this is just a subset of the non-oneM2M proximal IoT function. These are only data that are being actually exchanged. The oneM2M client application is not able to gain additional information that are linked to an incoming "Observation" like "Location" or "Sensor". This kind of information would need to be exchanged upfront in the configuration phase described in clause 6.3.

<mark>The following text is to be used when appropriate:</mark>

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