|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS#39 |
| Source:\* | Hyundai Mortors and KETI |
| Date:\* | 2019-02-17 |
| Contact:\* | JaeSeung Song, KETI, jssong@sejong.ac.kr  Lewis Nkenyereye, KETI, [nkenyele@sejong.ac.kr](mailto:nkenyele@sejong.ac.kr)  Youngjin Na, Hyundai Motors, [yjra@hyundai.com](mailto:yjra@hyundai.com)  Minbyeong Lee, Hyundai Motors, [minbyeong.lee@hyundai.com](mailto:minbyeong.lee@hyundai.com) |
| Reason for Change/s:\* | Introduce a concept of oneM2M message delivery repetition to optimize message delivery mechanism. |
| CR against: Release\* | R4 |
| CR against: WI\* | Active < Work Item number(optional)>  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  STE Small Technical Enhancements / < Work Item number(optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TR-0053 V0.2.1 |
| Clauses \* | Section 6.1.2 (Limitation of oneM2M messages), 6.1.3 (Potential Requirements), 6.1.4 (Potential Solutions) |
| 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 |
| Impacted other TS/TR(s) |  |
| 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 2017 (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

This contribution proposes optimizations to the oneM2M system to efficiently support message repetition.

Delivery of the same message multiple times to IoT devices and applications is one of common services used in many IoT platforms. Without this feature, IoT applications should support message repetition which requires additional service logic to maintaining states of message delivery, e.g., how many times delivered, what are the max repetition number.

Through supporting a message repetition feature in the oneM2M system, we can simplify the logic in IoT application and save message exchange between oneM2M System and IoT applications.

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

### 6.1.2 Limitations of oneM2M Messages

Request and response parameters as listed in **Table 6.1.1-1** and **Table 6.1.1-2** could make a oneM2M request or response message large. In the meantime, multiple consecutive request (or response) messages could contain the same request (or response) parameters with the same value; in another case, resource attributes contained in the “Content” parameter could be unexpected by the requestor or redundant in multiple consecutive messages. Such a large request or response message could be a burden for constrained IoT devices and/or networks with limited communication bandwidths. For example, the maximum layer-2 frame size for LoRaWAN at both US 915 MHz and EU 868 MHz band is 250 bytes, and the frame size in IEEE 802.15.4 networks is up to 127 bytes. The length of an oneM2M message if containing various request and response parameters could easily go beyond the maximum frame size which can be supported by the underlying Low-Power Wide Area Networks (LPWAN) like LoRaWAN or Low-Power Wireless Personal Area Networks (LoWPAN) such as IEEE 802.15.4.

**Figure 6.1.2-1** illustrates a smart metering use case, where each cellular UE smart meter uses low-power wide-area access technologies such as 3GPP Narrow-Band Internet of Things (NB-IoT) to communicate with a Server where an IoT service layer resides for storing and managing meter data from various UEs; the Server could be deployed by an electricity company. Basically, there could be a smart meter application running on each UE to periodically send meter readings to the Server. In addition, multiple smart meters (e.g. deployed in the same community) may report their readings to the Server in the same way (e.g. reporting frequency, how request messages shall be processed by the Server, etc.). As such, each smart meter may repeatedly send similar request messages to the Server, and multiple meters may also send similar request messages to the Server at different times. These two aspects are abstracted and discussed furthermore in **Figure 6.1.2-2** and **Figure 6.1.2-3**.



Figure 6.1.2-1: Smart Metering Use Case based on Cellular IoT

**Figure 6.1.2-2** illustrates interactions between an application and a service layer. In this example, the application (e.g. smart meter application on a smart meter in **Figure 6.1.2-1**) repeatedly sends request messages to the service layer. Each request message contains a set of request parameters; likewise, the corresponding response message contains response parameters. In addition, the application may request the same services/resources from the service layer during certain time durations; thus, each repeated request message includes the same set of request parameters.

**Figure 6.1.2-3** shows another example, where multiple applications (e.g. smart meter applications on smart meters in **Figure 6.1.2-1**) interact with the same service layer. In this scenario, although three (or more) applications could access different service/resources, they may instruct the service layer to process their request messages in the same way. For example, they may indicate to the service layer: the same request message expiration time, the same result expiration time, etc. Therefore, the request message from each application may contain the same set of request parameters.

Figure 6.1.2-x shows another scenario where applications interact with a service layer (we assume in this case smart vehicles that interact with infrastructures). In this example, an emergency service authority wants to send a safety warning message to a number of vehicles that are nearby an accident location. Because of several reasons (e.g., different vechiles continue approaching the accident location that have yet to receive the warning message) the emergency service authority wants to send the same warning message repeatedly. Therefore, message repetition is an indispensable feature for the oneM2M system.



Figure 6.1.2-x: Message repetition use case

The following limitations of oneM2M communication flow for constrained IoT devices are identified:

1. If a oneM2M request (or response) message contains too many request (or response) parameters, it cannot be sent in one layer-2 frame, which causes high message overhead and increases message transmission latency.
2. If a oneM2M request (or response) message is contained in multiple layer-2 frames, the loss of one layer-2 frame will results in the failure of oneM2M message delivery.
3. Even if a oneM2M request (or response) message can be completely contained in one layer-2 frame, the large size of the oneM2M message increases the message loss probability over the wireless channel.
4. Parameters contained in multiple consecutive oneM2M request (or response) messages could be the same and redundant, which increases message overhead unnecessarily.
5. The resource representation in the “Content” parameter may contain some extra attributes which are not expected or required by the requestor.
6. The resource representation in the “Content” parameter from multiple consecutive oneM2M request (or response) messages may contain the same or redundant attributes, which causes extra message overhead unnecessarily.
7. The oneM2M service layer does not currently support any compression mechanisms and is unable to leverage existing header compression protocols (e.g. IETF RFC 3095 for robust header compression) which have been designed for network layer, transport layer, and application protocol layer.. None of these are directly applicable to the oneM2M service layer.
8. If an oneM2M application wants to deliver the same oneM2M message repeatedly to oneM2M devices, it has to send multiple repeated oneM2M messages to the oneM2M service layer, which causes high message overhead and increases message transmission latency. This also increases the complexity of oneM2M Application logic.

Editor’s note: Security aspect should be considred to distinguish message repetition from a trusted party and unknown attackers.

Editor’s note: A mechanism for acknowledgement should be considered.

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

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

### 6.1.3 Potential Requirements

1. The oneM2M System shall support suitable request/response message interaction between a service layer and a constrained IoT device with low latency.
2. The oneM2M System shall support suitable request/response message interaction between a service layer and a constrained IoT device with low communication overhead.
3. The oneM2M System shall support suitable approaches for constrained IoT device to minimize request message size.
4. The oneM2M System shall support suitable approaches for constrained IoT device to minimize response message size.
5. The oneM2M System shall support suitable approaches for constrained IoT device to remove unrequired or redundant attributes from the resource representation as contained in the “Content” parameter.
6. The oneM2M System shall support suitable approaches for sending repetitive messages based on some specified conditions (e.g. schedule) to minimize message exchange overhead between oneM2M applications and the oneM2M System.

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