|  |
| --- |
| CHANGE REQUEST |
| Meeting ID:\* | SDS#41 |
| Source:\* | Hyundai Mortors and KETI |
| Date:\*  | 2019-07-01 |
| Contact:\* | JaeSeung Song, KETI, jssong@sejong.ac.krYoungjin Na, Hyundai Motor, yjra@hyundai.comMinbyeong Lee, Hyundai Motor, minbyeong.lee@hyundai.com  |
| Reason for Change/s:\* | Suggest a solution for oneM2M message delivery repetition using a dedicated resource.  |
| CR against: Release\* | R4  |
| CR against: WI\* | [x]  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 V0.3.0 |
| Clauses \* | 6.1.4 (Potential Solutions) |
| Type of change: \* | [ ]  Editorial change[ ]  Bug Fix or Correction[ ]  Change to existing feature or functionality[x]  New feature or functionalityOnly ONE of the above shall be ticked |
| Impacted other TS/TR(s) |  |
| 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 [x]  |
| 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 a potential solution to the oneM2M system to efficiently support message repetition.

The general concept of message repetition is to allow the delivery of the same message multiple times to IoT devices and applications as a common service via oneM2M IoT platform.

The proposing solution suggest to created a dedicated resource called <msgRepetition> to be used to generate predefined values. Once this new resource is created a target resource can be operated based on given information such as interval, duration, response handling policy.

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

6.1.4 Potential Solutions

*Editor’s Note: The section describes potential solutions related to optimizing/enhancing the oneM2M function to address the identified limitations and requirements.*

6.1.4.x Potential solution for message repetition using a dedicated resource

Message repetition is to configure a Hosting CSE to perform a specified operation repeatedly. For example, if a temperature sensor measures the same value (e.g., 25 celsius) most of the time, a dedicated resource representing message repetition can be created in the Hosting CSE to perform message repetition. In this case, the resource can hold information about the target resource in the Hosting CSE, an operation to performed to the target resource, a value to be used, a response policy and an interval of the message repetition. The Hosting CSE typically holds multiple message repetition requests so that it provides a resource to manage multiple message repetition requests.

Two new resources <*msgRepetitionList*> and <msgRepetition> are proposed to support oneM2M message repetition:

* <*msgRepetitionList*>: This is a resource to hold all the message repetition resources
* <*msgRepetition*>: This is a resource to hold all the information to perform message repetition. In order to support various message repetition in different circumstances, this resource contains attributes defined in Table 6.1.4.x-2.

|  |  |
| --- | --- |
|  |  |

**Figure 6.1.4.x-1: Structure of *<msgRepetitionList> and <msgRepetition>* resources**

Table 6.1.4.x-1: Proposed <*msgRepetitionList*> child resources

| Child Resource Type | Multiplicity | Description |
| --- | --- | --- |
| *<msgRepetition>* | 0..n | Achild resource that contains individual message repetition related information that defines how a message repetition needs to be performed.  |
| *<semanticDescriptor>* | 0..1 | Achild resource that contains semantic metadata to describe the parent <*msgRepetitionList*> resource. |

Table 6.1.4.x-2: Proposed <*msgRepetitionList*> attribute

| Attributes of <msgRepetitionList> | Description |
| --- | --- |
| *<numberOfMsgRepetition>* | Used to track the number of message repetition records for the purpose of management.  |

Table 6.1.4.x-3: Proposed <*msgRepetition*> attribute

| Attributes of <msgRepetition> | Description |
| --- | --- |
| *targetResource* | Used to indicate which resource is the target resource for the messge repetition. (e.g., *contentInstance*) |
| *targetOperation* | Used to indicate which operation needs to be performed on the resource addressed in the *targetResource* attribute. An example is CREATE and UPDATE.  |
| *repetitionInterval* | Used by a requester to set up an interval of message repetition. The unit of this attribute is second. An example interverl is 10 seconds. |
| *repetitionDuration* | Used by a requester to set up the duration of message repetition. The format of this attribute is date. An example of duration is 2030-01-01. Then the Hosting CSE performs message repetition until 2020-01-01. After the completion of message repetition until the given duration date, the associated <*msgRepetition*> resource is removed.  |
| *valueForRepetition* | Used by a requester to set up a value to be repeated (e.g., temperature 15 celsius).  |
| *nextForcedValue* | In many cases, a different value can be measured from a requester. Then the requester needs to force the meadured different value to be operated instead of planned message repetition based on the predefined value in valueForRepttition. This attribute defines the value to be operated for the next message repetition. This value is different from the value in the *valueForRepetition* attribute. This value becomes ‘None’ after the value is operated by the Hosting CSE, so that the Hosting CSE can continue message repetition based on the predefined information afterward.  |
| *responseMode* | This attribute defines a response handling policy. This attribute can be configured with the values such as ONCE, PER\_REPETITION, ACCUMULATED. Based on the configured value, the Hosting CSE responses to the requester. If the configured value is ACCUMULATED, the Hosting CSE includes all the message repetition information since the value in the lastResponse attribute.  |
| *repetitionOriginator* | This attribute defines the originator of the message repetition.  |
| *lastResponse* | This attribute contain Information about the last response sent to the Originator.  |

The following figure shows a possible procedure for performing the message delivery repetition.



Figure 6.1.4.x-1. Call flow for message repetition

* **Step 1**: AE#1 sends a request to create a resource (i.e., <*msgRepetition*>) for repeating intended message with the following information:
	+ For example, CREATE <*contentInstance*> under <*container-x*> with value ‘y’ until the given date ‘d’ for every time ‘t’.
	+ AE#1 can also configure a response policy, for example, no response is required.
* **Step 2**: IN-CSE then checks the access control policy of AE#1 to the target resource, <*container-x*>. If AE#1 has an access to create a resource under the <container-x> resource, IN-CSE creates <*msgRepetition*> under <*msgRepetitionList*> with the given information.
* **Step 3**: IN-CSE responses to AE#1 for the successful creation of <msgRepetition>.
* **Step 4**: AE#2 subscribes to <*container-x*> to be notified when a new <*contentInstance*> resource is created under <*container-x*>.
* **Step 5**: After time ‘t’ (e.g., 20 seconds) duration, IN-CSE checks the existence of <*msgRepetition*> related to <*container-x*>, then performs the requested message repetition. In this case, IN-CSE creates <*contentInstance*> resource with the value ‘y’ under the <*container-x*> resource.
* **Step 6**: AE#2 gets a notification for the creation of the new <*contentInstance*> under <*container-x*>.
* **Step 7**: Before reaching the next time ‘t’, AE#1 measures a value which is different (e.g., ‘z’) from the preconfigured value in the <*msgRepetition*> resource. As this value is not the value AE#1 is expecting, AE#1 updates the attribute (i.e., *nextForcedValue*) of <*msgRepetition*> about this exceptional value.
* **Step8**: IN-CSE updates the *nextForcedValue* attribute with the received value ‘z’. At reaching the next time ‘t’, IN-CSE performs CREATE operation of <*contentInstance*> under <*container-x*>. But this time, the value of <*contentInstance*> is the one configured in the nextForcedValue attribute ‘z’. Then IN-CSE reset the *nextForcedValue* attribute.
* **Step 9**: AE#2 gets a notification for the creation of the new <*contentInstance*> with the value ‘z’ under <*container-x*>.