|  |
| --- |
|  |

|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS #53 |
| Source:\* | Peter Niblett, IBM |
| Date:\* | 2021-02-16 |
| Reason for Change/s:\* | Clarify expected behaviour if non-confirmable messages are used (R3) |
| CR against: Release\* | Release 3 |
| CR against: WI\* | Active WI-xxxx  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0008 v.3.6.0 |
| Clauses \* | Modified clauses: 3, 5.1, 6.2.1, 6.3.0, 6.3.1, 6.3.2, 6.3.3, 6.3.4 |
| 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 CR is a mirror of SDS-2020-0374R05-Allow\_non-confirmable\_messages\_in\_CoAP. All changes are identical.

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of change 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 6.3.0 Introduction

This clause describes the behaviour of the CoAP layer depending on the ***Response Type*** parameter. Note that the CoAP messaging model defined in [1] applies to all message exchanges.

oneM2M Requests should be sent as CoAP Confirmable (CON) messages, although an Originator can send a request as a Non-confirmable (NON) message if there is a good reason for doing this. An Originator that relies on getting a response to its request should a Confirmable rather than a Non-confirmable message for its request.

oneM2M Responses should be sent as CoAP CON messages, although there is one case where a NON message should be used. This is indicated in clause 6.3.1.

If the Originator or Receiver sends a CON message it shall retransmit that message if it does not receive a CoAP acknowledgement message, as required by [1]. The recipient (Receiver or Originator) shall take care to de-duplicate CON messages as described in [1].

The recipient of a CoAP message shall process the oneM2M request or response it contains, even if it was sent as Non-confirmable.

### 6.3.1 Blocking case

1. If ***Response Type*** parameter is configured as "blockingRequest" (blocking case), the Originator (CoAP client) shall send the oneM2M request to the Receiver (CoAP server). The oneM2M ***Operation*** parameter shall be mapped to a CoAP Method according to Table 6.2.1-1.
2. After processing the oneM2M request, the Receiver shall send the oneM2M response in a CoAP response with a CoAP response code as given by Table 6.2.4-1. If the request was sent as a CoAP Confirmable message, the Receiver may either piggyback this response to the request on the CoAP ACK message, or send the response as a separate CoAP Confirmable message after it has sent the CoAP ACK. If the oneM2M request was sent in a Non-confirmable message, the oneM2M response shall be returned as a separate CoAP message. This response should be sent as a Non-confirmable CoAP message but it may be sent as Confirmable (this means that a receiver can, if it so chooses, send all Responses as Confirmable regardless of how the Request was sent).
3. The Originator’s CoAP binding may generate a response primitive containing a oneM2M ***Response Status Code*** of "REQUEST\_TIMEOUT" if it considers that it has taken too long for the CoAP response to come back from the Receiver. It shall ignore any response to the original request that it might receive after it has done this.

### 6.3.2 Non-Blocking Asynchronous case

1) If the ***Response Type*** parameter is configured as "nonBlockingRequestAsynch" (non-blocking asynchronous case), the Originator (CoAP client) should send the oneM2M request to the Receiver (CoAP server) as a CoAP Confirmable message. This request shall be sent using a CoAP POST method, and shall include the ***Operation*** parameter, mapped as described in clause 6.2.2.3.

2) The Receiver, after validating the request and before processing it fully, shall return a oneM2M response to the originator. It may either piggyback (2a) this response on the CoAP ACK message (if the request was sent as a Confirmable message), or send the response as a separate CoAP message after it has sent the CoAP ACK (2b). In this latter case it shall send the response as a Confirmable message.

* If the Receiver supports the <request> resource type, it shall respond with a 2.01 (Created) CoAP response code and a oneM2M ***Response Status Code*** of "ACCEPTED for nonBlockingRequestAsynch". The response shall include the URI of the new <request> resource in a sequence of one or more Location-Path and/or Location-Query Options.
* If the Receiver does not support the <request> resource type, it shall respond with a 2.04 (Changed) CoAP response code and a oneM2M ***Response Status Code*** of "ACCEPTED for nonBlockingRequestAsynch".

3) The Receiver, upon successful processing of the request, shall send a new CoAP Confirmable request message using POST method. This message contains a oneM2M NOTIFY primitive whose content contains the response to the original request.

4) The Originator may either piggyback a response to this request (4a) or send it as a separate CoAP response after the acknowledgment message (4b). This response shall contain the appropriate CoAP response code as defined in table 6.2.4-1 and have an empty payload.



Figure 6.3.2-1: Non-Blocking Asynchronous Case

### 6.3.3 Non-Blocking Synchronous case

1) If the ***Response Type*** parameter is configured as "nonBlockingRequestSynch" (non-blocking synchronous case), the Originator (CoAP client) should send the oneM2M request to the Receiver (CoAP server) as a CoAP Confirmable message. This request shall be sent using a CoAP POST method, and shall include the ***Operation*** parameter, mapped as described in clause 6.2.2.3.

2) The Receiver, after validating the request and before processing it fully, shall return a oneM2M response to the originator. It may either piggyback this response (2a) on the CoAP ACK message (if the request was sent as a CON message) or send the response as a separate CoAP message after it has sent the CoAP ACK (2b). In this latter case it shall send the response as a Confirmable message.

* If the Receiver supports the <request> resource type, it shall respond with a 2.01 (Created) CoAP response code and a oneM2M ***Response Status Code*** of "ACCEPTED for nonBlockingRequestSynch". The response shall include the URI of the new <request> resource in a sequence of one or more Location-Path and/or Location-Query Options.
* If the Receiver does not support the <request> resource type, it shall respond with a 5.01 (Not implemented) CoAP response code and a oneM2M ***Response Status Code*** of "NON\_BLOCKING\_SYNCH\_REQUEST\_NOT\_SUPPORTED".

3) The Originator can use the <request> resource reference to synchronously retrieve the <request> resource that contains the response to the original request.

4) The Receiver, upon receipt of this retrieve request, shall handle it as in clause 6.3.1 since it is a non-blocking request.

NOTE: If the Receiver is a Transit CSE, the Receiver acts as CoAP client and CoAP server.



Figure 6.3.3-1: Non-Blocking Synchronous Case

### 6.3.4 Flex Blocking case

1) If the ***Response Type*** parameter is configured as "flex blocking", the Originator (CoAP client) should send the oneM2M request to the Receiver (CoAP server) as a CoAP Confirmable message. This request shall be sent using a CoAP POST method, and shall include the ***Operation*** parameter, mapped as described in clause 6.2.2.3.

2) The Receiver shall determine whether to handle the request using "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" mode:

* If the Receiver chooses "nonBlockingRequestAsynch" processing proceeds as described in clause 6.3.2, starting from step 2).
* If the Receiver chooses "nonBlockingRequestSynch" processing proceeds as described in clause 6.3.3, starting from step 2).

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of Change 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 5.1 Required Features

This clause explicitly specifies the required features of the CoAP layer for oneM2M to properly bind oneM2M primitives into CoAP messages:

* The 4-byte binary CoAP message header is defined in section 3 of IETF RFC 7252 [1].
* Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK) and Reset (RST) messages shall be supported. The Reset message is used to send an error message in response to a malformed Confirmable message in CoAP layer.
* GET, PUT, POST and DELETE methods shall be supported. oneM2M primitives map to these methods.
* The CoAP Response Codes specified in clause 6.2.4 shall be supported for oneM2M ***Response Status Code*** parameter mapping.
* The Uri-Host, Uri-Port, Uri-Path, and Uri-Query shall be supported.
* The Content-Type Option shall be used to indicate the media types of the payload.
* Block-wise transfers feature may be supported to carry large payloads.
* The Caching feature may be supported.

## 5.2 Introduction to CoAP

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of Change 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 6.2.1 Header

This clause specifies how to configure CoAP header information:

* The Version field shall be configured as 1.
* The Type field shall be configured according to clause 6.3. The Reset message is used to indicate an error in response to a malformed message in CoAP layer.
* In case of a request, the Code field indicates the CoAP Method. If the oneM2M operation is sent as a Blocking request the oneM2M ***Operation*** parameter shall be mapped to a CoAP Method according to the table 6.2.1-1. In non-blocking and flex blocking cases, the request shall use the CoAP POST method, and the Operation parameter shall be mapped as described in clause 6.2.2.3.
* In case of a response, the Code field indicates the CoAP Response Code. The oneM2M ***Response Status Code*** parameter shall be mapped to a CoAP Response Code as specified in clause 6.2.4.
* The Originator and Receiver shall set the 16 bit MessageId in accordance with the CoAP specification [1] and shall retry transmission of all unacknowledged Confirmable messages, as required by that specification.

Table 6.2.1-1: oneM2M Operation Parameter Mapping

|  |  |  |
| --- | --- | --- |
| oneM2M Operation Parameter | CoAP Method | CoAP Method Code |
| CREATE | POST | 0.02 |
| RETRIEVE | GET | 0.01 |
| UPDATE | PUT | 0.03 |
| DELETE | DELETE | 0.04 |
| NOTIFY | POST | 0.02 |

At the Receiver, a CoAP request message with a POST method that does not carry an ***Operation*** parameter shall be mapped to a oneM2M CREATE or NOTIFY operation in accordance with the existence of the ***Resource Type***.

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of Change 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 3 Abbreviations and acronyms

For the purposes of the present document, the following abbreviations and acronyms apply:

ACK CoAP Acknowledgement message

AE Application Entity

CON CoAP Confirmable message

CSE Common Service Entity

DTLS Datagram Transport Layer Security

HTTP Hyper Text Transfer Protocol

IANA Internet Assigned Numbers Authority

IP Internet Protocol

NON CoAP Non-confirmable message

RST CoAP ReSeT message

TCP Transport Control Protocol

TLS Transport Layer Security

TLV Tag - Length - Value (data structure)

UDP User Datagram Protocol

URI Uniform Resource Identifier

XML eXtensible Markup Language

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*