CORE
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2021
October 29, 2020

M. Boucadair
Orange
J. Shallow

Constrained Application Protocol (CoAP) Block-Wise Transfer Options for Faster Transmission

draft-ietf-core-new-block-02

Abstract

This document specifies alternative Constrained Application Protocol (CoAP) Block-Wise transfer options: Q-Block1 and Q-Block2 Options.

These options are similar to the CoAP Block1 and Block2 Options, not a replacement for them, but do enable faster transmission rates for large amounts of data with less packet interchanges as well as supporting faster recovery should any of the blocks get lost in transmission.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Existing CoAP Block-Wise Transfer Options

The Constrained Application Protocol (CoAP) [RFC7252], although inspired by HTTP, was designed to use UDP instead of TCP. The message layer of CoAP over UDP includes support for reliable delivery, simple congestion control, and flow control. [RFC7959] introduced the CoAP Block1 and Block2 Options to handle data records that cannot fit in a single IP packet, so not having to rely on IP fragmentation and further updated by [RFC8323] for use over TCP, TLS, and Websockets.

The CoAP Block1 and Block2 Options work well in environments where there are no or minimal packet losses. These options operate synchronously where each block has to be requested and can only ask for (or send) the next block when the request for the previous block has completed. Packet, and hence block transmission rate, is controlled by Round Trip Times (RTTs).

There is a requirement for these blocks of data to be transmitted at higher rates under network conditions where there may be asymmetrical transient packet loss. An example is when a network is subject to a Distributed Denial of Service (DDoS) attack and there is a need for DDoS mitigation agents relying upon CoAP to communicate with each other (e.g., [I-D.ietf-dots-telemetry]). As a reminder, [RFC7959] recommends use of Confirmable (CON) responses to handle potential packet loss; which does not work with a flooded pipe DDoS situation.

1.2. Alternative CoAP Block-Wise Transfer Options

This document introduces the CoAP Q-Block1 and Q-Block2 Options. These options are similar in operation to the CoAP Block1 and Block2 Options respectively, they are not a replacement for them, but have the following benefits:

o They can operate in environments where packet loss is highly asymmetrical.

o They enable faster transmissions of sets of blocks of data with less packet interchanges.

o They support faster recovery should any of the Blocks get lost in transmission.

o They support sending an entire body using Non-confirmable (NON) without requiring a response from the peer.

There are the following disadvantages over using CoAP Block 1 and Block2 Options:

- o Loss of lock-stepping so payloads are not always received in the correct (block ascending) order.

- o Additional congestion control measures need to be put in place.

Using NON messages, the faster transmissions occur as all the Blocks can be transmitted serially (as are IP fragmented packets) without having to wait for an acknowledgement or next request from the remote CoAP peer. Recovery of missing Blocks is faster in that multiple missing Blocks can be requested in a single CoAP packet. Even if there is asymmetrical packet loss, a body can still be sent and received by the peer whether the body compromises of a single or multiple payloads assuming no recovery is required.

Note that the same performance benefits can be applied to Confirmable messages if the value of NSTART is increased from 1 (Section 4.7 of [RFC7252]). However, the asymmetrical packet loss is not a benefit here. Some sample examples with Confirmable messages are provided in Appendix A.

There is little, if any, benefit of using these options with CoAP running over a reliable connection [RFC8323]. In this case, there is no differentiation between Confirmable and NON as they are not used.

A CoAP endpoint can acknowledge all or a subset of the blocks. Concretely, the receiving CoAP endpoint informs the CoAP endpoint sender either successful receipt or reports on all blocks in the body that have been not yet been received. The CoAP endpoint sender will then retransmit only the blocks that have been lost in transmission.

Q-Block1 and Q-Block2 Options can be used instead of Block1 and Block2 Options respectively when the different transmission semantics are required. If the option is not supported by a peer, then transmissions can fall back to using Block1 and Block2 respectively.

The deviations from Block1 and Block2 Options are specified in Section 3. Pointers to appropriate [RFC7959] sections are provided.

The specification refers to the base CoAP methods defined in Section 5.8 of [RFC7252] and the new CoAP methods, FETCH, PATCH, and iPATCH introduced in [RFC8132].

1.3. Updated CoAP Response Code (4.08)

This document updates the 4.08 (Request Entity Incomplete) by defining an additional message format for reporting on payloads using the Q-Block1 Option that are not received by the server.

See Section 4 for more details.

1.4. Applicability Scope

The block-wise transfer specified in [RFC7959] covers the general case, but falls short in situations where packet loss is highly asymmetrical. The mechanism specified in this document provides roughly similar features to the Block1/Block2 Options. It provides additional properties that are tailored towards the intended use case. Concretely, this mechanism primarily targets applications such as DDoS Open Threat Signaling (DOTS) that can't use Confirmable (CON) responses to handle potential packet loss and that support application-specific mechanisms to assess whether the remote peer is able to handle the messages sent by a CoAP endpoint (e.g., DOTS heartbeats in Section 4.7 of [RFC8782]).

The mechanism includes guards to prevent a CoAP agent from overloading the network by adopting an aggressive sending rate. These guards MUST be followed in addition to the existing CoAP congestion control as specified in Section 4.7 of [RFC7252]. See Section 6 for more details.

This mechanism is not intended for general CoAP usage, and any use outside the intended use case should be carefully weighed against the loss of interoperability with generic CoAP applications. It is hoped that the experience gained with this mechanism can feed future extensions of the block-wise mechanism that will both generally applicable and serve this particular use case.

It is not recommended that these options are used in a NoSec security mode (Section 9 of [RFC7252]) as the source endpoint needs to be trusted.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers should be familiar with the terms and concepts defined in [RFC7252].

The terms "payload" and "body" are defined in [RFC7959]. The term "payload" is thus used for the content of a single CoAP message (i.e., a single block being transferred), while the term "body" is

used for the entire resource representation that is being transferred in a block-wise fashion.

3. The Q-Block1 and Q-Block2 Options

3.1. Properties of Q-Block1 and Q-Block2 Options

The properties of Q-Block1 and Q-Block2 Options are shown in Table 1. The formatting of this table follows the one used in Table 4 of [RFC7252] (Section 5.10). The C, U, N, and R columns indicate the properties Critical, Unsafe, NoCacheKey, and Repeatable defined in Section 5.4 of [RFC7252]. Only C and U columns are marked for the Q-Block1 Option. C, U, and R columns are marked for the Q-Block2 Option.

Number	C	U	N	R	Name	Format	Length	Default
TBA1	x	x			Q-Block1	uint	0-3	(none)
TBA2	x	x		x	Q-Block2	uint	0-3	(none)

Table 1: CoAP Q-Block1 and Q-Block2 Option Properties

The Q-Block1 and Q-Block2 Options can be present in both the request and response messages. The Q-Block1 Option pertains to the request payload and the Q-Block2 Option pertains to the response payload. The Content-Format Option applies to the body, not to the payload (i.e., it must be the same for all payloads of the same body).

Q-Block1 Option is useful with the payload-bearing POST, PUT, PATCH, and iPATCH requests and their responses (2.01 and 2.04).

Q-Block2 Option is useful with GET, POST, PUT, FETCH, PATCH, and iPATCH requests and their payload-bearing responses (2.01, 2.03, 2.04, and 2.05) (Section 5.5 of [RFC7252]).

A CoAP endpoint (or proxy) MUST support either both or neither of the Q-Block1 and Q-Block2 Options.

To indicate support for Q-Block2 responses, the CoAP client MUST include the Q-Block2 Option in a GET or similar request, the Q-Block2 Option in a PUT or similar request, or the Q-Block1 Option in a PUT or similar so that the server knows that the client supports this Q-Block2 functionality should it need to send back a body that spans multiple payloads. Otherwise, the server would use the Block2 Option (if supported) to send back a message body that is too large to fit into a single IP packet [RFC7959].

If Q-Block1 Option is present in a request or Q-Block2 Option in a response (i.e., in that message to the payload of which it pertains), it indicates a block-wise transfer and describes how this specific block-wise payload forms part of the entire body being transferred. If it is present in the opposite direction, it provides additional control on how that payload will be formed or was processed.

Implementation of the Q-Block1 and Q-Block2 Options is intended to be optional. However, when it is present in a CoAP message, it MUST be processed (or the message rejected). Therefore, Q-Block1 and Q-Block2 Options are identified as Critical options.

The Q-Block1 and Q-Block2 Options are unsafe to forward. That is, a CoAP proxy that does not understand the Q-Block1 (or Q-Block2) Option MUST reject the request or response that uses either option.

The Q-Block2 Option is repeatable when requesting re-transmission of missing Blocks, but not otherwise. Except that case, any request carrying multiple Q-Block1 (or Q-Block2) Options MUST be handled following the procedure specified in Section 5.4.5 of [RFC7252].

The Q-Block1 and Q-Block2 Options, like the Block1 and Block2 Options, are both a class E and a class U in terms of OSCORE processing (see Section 4.1 of [RFC8613]): The Q-Block1 (or Q-Block2) Option MAY be an Inner or Outer option. The Inner and Outer values are therefore independent of each other. The Inner option is encrypted and integrity protected between clients and servers, and provides message body identification in case of end-to-end fragmentation of requests. The Outer option is visible to proxies and labels message bodies in case of hop-by-hop fragmentation of requests.

3.2. Structure of Q-Block1 and Q-Block2 Options

The structure of Q-Block1 and Q-Block2 Options follows the structure defined in Section 2.2 of [RFC7959].

There is no default value for the Q-Block1 and Q-Block2 Options. Absence of one of these options is equivalent to an option value of 0 with respect to the value of block number (NUM) and more bit (M) that could be given in the option, i.e., it indicates that the current block is the first and only block of the transfer (block number is set to 0, M is unset). However, in contrast to the explicit value 0, which would indicate a size of the block (SZX) of 0, and thus a size value of 16 bytes, there is no specific explicit size implied by the absence of the option -- the size is left unspecified. (As for any uint, the explicit value 0 is efficiently indicated by a zero-length

option; this, therefore, is different in semantics from the absence of the option).

3.3. Using the Q-Block1 Option

The Q-Block1 Option is used when the client wants to send a large amount of data to the server using the POST, PUT, PATCH, or iPATCH methods where the data and headers do not fit into a single packet.

When Q-Block1 Option is used, the client MUST include a single Request-Tag Option [I-D.ietf-core-echo-request-tag]. The Request-Tag value MUST be the same for all of the blocks in the body of data that is being transferred. It is also used to identify a particular block of a body that needs to be re-transmitted. The Request-Tag is opaque in nature, but it is RECOMMENDED that the client treats it as an unsigned integer of 8 bytes in length. An implementation may want to consider limiting this to 4 bytes to reduce packet overhead size. The server still treats it as an opaque entity. The Request-Tag value MUST be different for distinct bodies or sets of blocks of data and SHOULD be incremented whenever a new body of data is being transmitted for a CoAP session between peers. The initial Request-Tag value SHOULD be randomly generated by the client.

For Confirmable transmission, the server MUST continue to acknowledge each packet. NSTART will also need to be increased from the default (1) to get faster transmission rates.

Each individual payload of the body is treated as a new request (see Section 5).

A 2.01 (Created) or 2.04 (Changed) Response Code indicates successful receipt of the entire body.

The 2.31 (Continue) Response is not used in the current version of the specification.

A 4.00 (Bad Request) Response Code MUST be returned if the request does not include a Request-Tag Option but does include a Q-Block1 option.

A 4.02 (Bad Option) Response Code MUST be returned if the server does not support the Q-Block1 Option.

A 4.13 (Request Entity Too Large) Response Code can be returned under similar conditions to those discussed in Section 2.9.3 of [RFC7959].

A 4.08 (Request Entity Incomplete) Response Code returned without Content-Type "application/missing-blocks+cbor-seq" (Section 10.2) is handled as in Section 2.9.2 [RFC7959].

A 4.08 (Request Entity Incomplete) Response Code returned with Content-Type "application/missing-blocks+cbor-seq" indicates that some of the payloads are missing and need to be resent. The client then re-transmits the missing payloads using the Request-Tag and Q-Block1 to specify the block number, SZX, and M bit as appropriate. The Request-Tag value to use is determined from the payload of the 4.08 (Request Entity Incomplete) Response Code. If the client does not recognize the Request-Tag, the client can ignore this response.

If the server has not received all the payloads of a body, but one or more payloads have been received, it SHOULD wait for up to MAX_TRANSMIT_SPAN (Section 4.8.2 of [RFC7252]) before sending the 4.08 (Request Entity Incomplete) Response Code. However, this time MAY be reduced to two times ACK_TIMEOUT before sending a 4.08 (Request Entity Incomplete) Response Code to cover the situation where MAX_PAYLOADS has been triggered by the client causing a break in transmission.

If the client transmits a new body of data with a new Request-Tag to the same resource on a server, the server MUST remove any partially received body held for a previous Request-Tag for that resource.

If the server receives a duplicate block with the same Request-Tag, it SHOULD silently ignore the packet.

A server SHOULD only maintain a partial body (missing payloads) for up to EXCHANGE_LIFETIME (Section 4.8.2 of [RFC7252]).

3.4. Using the Q-Block2 Option

In a request for any block number, the M bit unset indicates the request is just for that block. If the M bit is set, this indicates that this is a request for this block and for all of the remaining blocks within the body. If the server receives multiple requests (implied or otherwise) for the same block, it MUST only send back one instance of that block.

The payloads sent back from the server as a response MUST all have the same ETag (Section 5.10.6 of [RFC7252]) for the same body. The server MUST NOT use the same ETag value for different representations of a resource.

The ETag is opaque in nature, but it is RECOMMENDED that the server treats it as an unsigned integer of 8 bytes in length. An

implementation may want to consider limiting this to 4 bytes to reduce packet overhead size. The client still treats it as an opaque entity. The ETag value MUST be different for distinct bodies or sets of blocks of data and SHOULD be incremented whenever a new body of data is being transmitted for a CoAP session between peers. The initial ETag value SHOULD be randomly generated by the server.

If the client detects that some of the payloads are missing, the missing payloads are requested by issuing a new GET, POST, PUT, FETCH, PATCH, or iPATCH request that contains one or more Q-Block2 Options that define the missing blocks with the M bit unset.

The requested missing block numbers MUST have an increasing block number in each additional Q-Block2 Option with no duplicates. The server SHOULD respond with a 4.00 (Bad Request) if this is the case.

The ETag Option MUST NOT be used in the request as the server could respond with a 2.03 (Valid Response) with no payload. If the server responds with a different ETag Option value (as the resource representation has changed), then the client SHOULD drop all the payloads for the current body that are no longer valid.

The client may elect to request the missing blocks or just ignore the partial body. It SHOULD wait for up to MAX_TRANSMIT_SPAN (Section 4.8.2 of [RFC7252]) before issuing a GET, POST, PUT, FETCH, PATCH, or iPATCH request for the missing blocks. However, this time MAY be reduced to two times ACK_TIMEOUT before sending the request to cover the situation where MAX_PAYLOADS has been triggered by the server causing a break in transmission.

With NON transmission, the client only needs to indicate that some of the payloads are missing by issuing a GET, POST, PUT, FETCH, PATCH, or iPATCH request for the missing blocks.

For Confirmable transmission, the client SHOULD continue to acknowledge each packet as well as issuing a separate GET, POST, PUT, FETCH, PATCH, or iPATCH for the missing blocks.

If the server transmits a new body of data (e.g., a triggered Observe) with a new ETag to the same client as an additional response, the client MUST remove any partially received body held for a previous ETag.

If the client receives a duplicate block with the same ETag, it SHOULD silently ignore the packet.

A client SHOULD only maintain a partial body (missing payloads) for up to EXCHANGE_LIFETIME (Section 4.8.2 of [RFC7252]) or as defined by the Max-Age Option whichever is the less.

If there is insufficient space to create a response PDU with a block size of 16 bytes (SZX = 0) to reflect back all the request options as appropriate, a 4.13 (Request Entity Too Large) is returned without the Size2 Option.

3.5. Working with Observe and Q-Block2 Options

As the blocks of the body are sent without waiting for acknowledgement of the individual blocks, the Observe value [RFC7641] MUST be the same for all the blocks of the same body.

If the client requests missing blocks, this is treated as a new request. The Observe value may change but MUST still be reported. If the ETag value changes then the previously received partial body should be destroyed and the whole body re-requested.

3.6. Working with Size1 and Size2 Options

Section 4 of [RFC7959] defines two CoAP options: Size1 for indicating the size of the representation transferred in requests and Size2 for indicating the size of the representation transferred in responses.

The Size1 or Size2 option values MUST exactly represent the size of the data on the body so that any missing data can easily be determined.

The Size1 Option MUST be used with the Q-Block1 Option when used in a request. The Size2 Option MUST be used with the Q-Block2 Option when used in a response.

If Size1 or Size2 Options are used, they MUST be used in all payloads of the body and MUST have the same value.

3.7. Use of Q-Block1 and Q-Block2 Options Together

The behavior is similar to the one defined in Section 3.3 of [RFC7959] with Q-Block1 substituted for Block1 and Q-Block2 for Block2.

4. The Use of 4.08 (Request Entity Incomplete) Response Code

4.08 (Request Entity Incomplete) Response Code has a new Content-Type "application/missing-blocks+cbor-seq" used to indicate that the

server has not received all of the blocks of the request body that it needs to proceed.

Likely causes are the client has not sent all blocks, some blocks were dropped during transmission, or the client has sent them sufficiently long ago that the server has already discarded them.

The data payload of the 4.08 (Request Entity Incomplete) Response Code is encoded as a CBOR Sequence [RFC8742]. First is CBOR encoded Request-Tag followed by 1 or more missing CBOR encoded missing block numbers. The missing block numbers MUST be unique in each 4.08 (Request Entity Incomplete) when created by the server; the client SHOULD drop any duplicates in the same 4.08 (Request Entity Incomplete) message.

The Content-Format Option (Section 5.10.3 of [RFC7252]) MUST be used in the 4.08 (Request Entity Incomplete) Response Code. It MUST be set to "application/missing-blocks+cbor-seq" (see Section 10.2).

The Concise Data Definition Language [RFC8610] for the data describing these missing blocks is as follows:

```
; This defines an array, the elements of which are to be used
; in a CBOR Sequence:
payload = [request-tag, + missing-block-number]
request-tag = bstr
; A unique block number not received:
missing-block-number = uint
```

Figure 1: Structure of the Missing Blocks Payload

If the size of the 4.08 (Request Entity Incomplete) response packet is larger than that defined by Section 4.6 [RFC7252], then the number of missing blocks MUST be limited so that the response can fit into a single packet. If this is the case, then the server can send subsequent 4.08 (Request Entity Incomplete) responses containing the missing blocks on receipt of a new request providing a missing payload with the same Request-Tag.

The missing blocks MUST be reported in ascending order without any duplicates. The client SHOULD silently drop 4.08 (Request Entity Incomplete) responses not adhering with this behavior.

Implementation Note: Updating the payload without overflowing the overall packet size as each block number can be of varying length needs consideration. It is possible to use Indefinite-Length Arrays (Section 2.2.1 of [RFC7049]), limit the array count to 23 (Undefined value) so that the array data byte can be updated with

the overall length once the payload length is confirmed or limited to MAX_PAYLOADS count. Limiting the count to MAX_PAYLOADS means that Congestion Control is less likely to be invoked on the server.

5. The Use of Tokens

Each new request MUST use a unique Token (Section 4 of [I-D.ietf-core-echo-request-tag]). Additional responses may use the same Token.

Implementation Note: To minimize on the number of tokens that have to be tracked by clients, it is recommended that the bottom 32 bits is kept the same for the same body and the upper 32 bits contains the individual payload number.

Servers continue to treat the token as a unique opaque entity. If an individual payload has to be resent (e.g., requested upon packet loss), then the retransmitted packet is treated as a new request (i.e., the bottom 32 bits must change).

6. Congestion Control

PROBING_RATE parameter in CoAP indicates the average data rate that must not be exceeded by a CoAP endpoint in sending to a peer endpoint that does not respond. The body of blocks will be subjected to PROBING_RATE (Section 4.7 of [RFC7252]).

Each NON 4.08 (Request Entity Incomplete) Response Codes is subjected to PROBING_RATE.

Each NON GET or similar request using Q-Block2 Option is subjected to PROBING_RATE.

As the sending of many payloads of a single body may itself cause congestion, it is RECOMMENDED that after transmission of every set of MAX_PAYLOADS payloads of a single body, a delay is introduced of ACK_TIMEOUT (Section 4.8.2 of [RFC7252]) before the next set of payload transmissions to manage potential congestion issues. MAX_PAYLOADS should be configurable with a default value of 10.

Note: The default value is chosen for reasons similar to those discussed in Section 5 of [RFC6928].

For NON transmissions, it is permissible, but not required, to send the ultimate payload of a MAX_PAYLOADS set as a Confirmable packet. If a Confirmable packet is used, then the transmitting peer MUST wait

for the ACK to be returned before sending the next set of payloads, which can be in time terms less than the ACK_TIMEOUT delay.

Also, for NON transmissions, it is permissible, but not required, to send a Confirmable packet for the final payload of a body transfer (that is, M bit unset). If a Confirmable packet is used, then the transmitting peer MUST wait for the appropriate response to be returned for successful transmission, or respond to requests for the missing blocks (if any).

The sending of the set of missing blocks is subject to MAX_PAYLOADS.

> Note: A delay of ACK_TIMEOUT after every transmission of MAX_PAYLOADS blocks may be observed even if the peer agent is able to handle more blocks without experiencing an overload. This delay can be reduced by using CON for the MAX_PAYLOADS packet to trigger sending the next set of data when the ACK is received. Nevertheless, this behavior is likely to create other timeout issues in a lossy environment (e.g., unidirectional loss as in DDoS pipe flooding). The use of NON is thus superior but requires an additional signal in the MAX_PAYLOADS packet to seek for a 2.31 (Continue) from the peer if it is ready to receive the next set of blocks.

7. Caching Considerations

Caching block based information is not straight forward in a proxy. For Q-Block1 and Q-Block2 Options, it is expected that the proxy will reassemble the body (using any appropriate recovery options for packet loss) before passing on the body to the appropriate CoAP endpoint. The onward transmission of the body does not require the use of the Q-Block1 or Q-Block2 Options as these options may not be supported in that link. This means that the proxy must fully support the Q-Block1 and Q-Block2 Options.

How the body is cached in the initial CoAP client (Q-Block1) or ultimate CoAP server (Q-Block2) is implementation specific.

As the entire body is being cached in the proxy, the Q-Block1 and Q-Block2 Options are not part of the cache key.

For Q-Block2 responses, the ETag Option value is associated with the data (and onward transmitted to the CoAP client), but is not part of the cache key.

For requests with Q-Block1 Option, the Request-Tag Option is associated with the build up of the body from successive payloads,

but is not part of the cache key. For the onward transmission of the body using CoAP, a new Request-Tag SHOULD be generated and used.

It is possible that two or more CoAP clients are concurrently updating the same resource through a common proxy to the same CoAP server using Q-Block1 (or Block1) Option. If this is the case, the first client to complete building the body causes that body to start transmitting to the CoAP server with an appropriate Request-Tag value. When the next client completes building the body, any existing partial body transmission to the CoAP server is terminated and the new body representation transmission starts with a new Request-Tag value.

A proxy that supports Q-Block2 Option MUST be prepared to receive a GET or similar message indicating one or more missing blocks. The proxy will serve from its cache the missing blocks that are available in its cache in the same way a server would send all the appropriate Q-Block2s. If the cache key matching body is not available in the cache, the proxy MUST request the entire body from the CoAP server using the information in the cache key.

How long a CoAP endpoint (or proxy) keeps the body in its cache is implementation specific (e.g., it may be based on Max-Age).

8. HTTP-Mapping Considerations

As a reminder, the basic normative requirements on HTTP/CoAP mappings are defined in Section 10 of [RFC7252]. The implementation guidelines for HTTP/CoAP mappings are elaborated in [RFC8075].

The rules defined in Section 5 of [RFC7959] are to be followed.

9. Examples of Selective Block Recovery

This section provides some sample flows to illustrate the use of Q-Block1 and Q-Block2 Options. Figure 2 lists the conventions that are used in the following subsections.

```
      T: Token value
      O: Observe Option value
      M: Message ID
     RT: Request-Tag
     ET: ETag
    QB1: Q-Block1 Option values NUM/More/SZX
    QB2: Q-Block2 Option values NUM/More/SZX
      \: Trimming long lines
   [[]]: Comments
   -->X: Message loss
   X<--: Message loss
```

Figure 2: Notations Used in the Figures

9.1. Q-Block1 Option: Non-Confirmable Example

Figure 3 depicts an example of a NON PUT request conveying Q-Block1 Option. All the blocks are received by the server.

```
        CoAP        CoAP
       Client      Server
         |           |
         +--------->| NON PUT /path M:0x01 T:0xf0 RT=10 QB1:0/1/1024
         +--------->| NON PUT /path M:0x02 T:0xf1 RT=10 QB1:1/1/1024
         +--------->| NON PUT /path M:0x03 T:0xf2 RT=10 QB1:2/1/1024
         +--------->| NON PUT /path M:0x04 T:0xf3 RT=10 QB1:3/0/1024
         |<---------+ NON 2.04 M:0xf1 T:0xf3
             ...
```

Figure 3: Example of NON Request with Q-Block1 Option (Without Loss)

Consider now a scenario where a new body of data is to be sent by the client, but some blocks are dropped in transmission as illustrated in Figure 4.

```
        CoAP        CoAP
       Client      Server
         |           |
         +--------->| NON PUT /path M:0x05 T:0xe0 RT=11 QB1:0/1/1024
         +--->X     | NON PUT /path M:0x06 T:0xe1 RT=11 QB1:1/1/1024
         +--->X     | NON PUT /path M:0x07 T:0xe2 RT=11 QB1:2/1/1024
         +--------->| NON PUT /path M:0x08 T:0xe3 RT=11 QB1:3/0/1024
         |           |
             ...
```

Figure 4: Example of NON Request with Q-Block1 Option (With Loss)

The server realizes that some blocks are missing and asks for the missing ones in one go (Figure 5). It does so by indicating which blocks have been received in the data portion of the response.

```
        CoAP        CoAP
       Client      Server
         |           |
             ...
         |<---------+ NON 4.08 M:0xf2 T:0xe3 [Missing 1,2 for RT=11]
         +--------->| NON PUT /path M:0x09 T:0xe4 RT=11 QB1:1/1/1024
         +--->X     | NON PUT /path M:0x0a T:0xe5 RT=11 QB1:2/1/1024
         |           |
         |<---------+ NON 4.08 M:0xf3 T:0xe4 [Missing 2 for RT=11]
         +--------->| NON PUT /path M:0x0b T:0xe6 RT=11 QB1:2/1/1024
         |<---------+ NON 2.04 M:0xf4 T:0xe6
         |           |
             ...
```

Figure 5: Example of NON Request with Q-Block1 Option (Blocks Recovery)

Under high levels of traffic loss, the client can elect not to retry sending missing blocks of data. This decision is implementation specific.

9.2. Q-Block2 Option: Non-Confirmable Example

Figure 6 illustrates the example of Q-Block2 Option. The client sends a NON GET carrying an Observe and a Q-Block2 Options. The Q-Block2 Option indicates a size hint (1024 bytes). This request is replied by the server using four (4) blocks that are transmitted to the client without any loss. Each of these blocks carries a Q-Block2 Option. The same process is repeated when an Observe is triggered, but no loss is experienced by any of the notification blocks.

```
        CoAP        CoAP
       Client      Server
         |           |
         +--------->| NON GET /path M:0x01 T:0xf0 O:0 QB2:0/0/1024
         |<---------+ NON 2.05 M:0xf1 T:0xf0 O:1234 ET=21 QB2:0/1/1024
         |<---------+ NON 2.05 M:0xf2 T:0xf0 O:1234 ET=21 QB2:1/1/1024
         |<---------+ NON 2.05 M:0xf3 T:0xf0 O:1234 ET=21 QB2:2/1/1024
         |<---------+ NON 2.05 M:0xf4 T:0xf0 O:1234 ET=21 QB2:3/0/1024
              ...
           [[Observe triggered]]
         |<---------+ NON 2.05 M:0xf5 T:0xf0 O:1235 ET=22 QB2:0/1/1024
         |<---------+ NON 2.05 M:0xf6 T:0xf0 O:1235 ET=22 QB2:1/1/1024
         |<---------+ NON 2.05 M:0xf7 T:0xf0 O:1235 ET=22 QB2:2/1/1024
         |<---------+ NON 2.05 M:0xf8 T:0xf0 O:1235 ET=22 QB2:3/0/1024
             ...
```

Figure 6: Example of NON Notifications with Q-Block2 Option (Without Loss)

Figure 7 shows the example of an Observe that is triggered but for which some notification blocks are lost. The client detects the missing blocks and request their retransmission. It does so by indicating the blocks that were successfully received.

```
          CoAP        CoAP
         Client      Server
           |           |
               ...
              [[Observe triggered]]
           |<---------+ NON 2.05 M:0xf9 T:0xf0 O:1236 ET=23 QB2:0/1/1024
           |     X<---+ NON 2.05 M:0xfa T:0xf0 O:1236 ET=23 QB2:1/1/1024
           |     X<---+ NON 2.05 M:0xfb T:0xf0 O:1236 ET=23 QB2:2/1/1024
           |<---------+ NON 2.05 M:0xfc T:0xf0 O:1236 ET=23 QB2:3/0/1024
           |           |
        [[Client realizes blocks are missing and asks for the missing
          ones in one go]]
           +--------->| NON GET /path M:0x02 T:0xf1 QB2:1/0/1024\
           |           |                             QB2:2/0/1024
           |     X<---+ NON 2.05 M:0xfd T:0xf1 ET=23 QB2:1/1/1024
           |<---------+ NON 2.05 M:0xfe T:0xf1 ET=23 QB2:2/1/1024
           |           |
        [[Get the final missing block]]
           +--------->| NON GET /path M:0x03 T:0xf2 QB2:1/0/1024
           |<---------+ NON 2.05 M:0xff T:0xf2 ET=23 QB2:1/1/1024
               ...
```

Figure 7: Example of NON Notifications with Q-Block2 Option (Blocks Recovery)

Under high levels of traffic loss, the client can elect not to retry getting missing blocks of data. This decision is implementation specific.

10. IANA Considerations

10.1. New CoAP Options

IANA is requested to add the following entries to the "CoAP Option Numbers" sub-registry [Options]:

Number	Name	Reference
TBA1	Q-Block1	[RFCXXXX]
TBA2	Q-Block2	[RFCXXXX]

Table 2: CoAP Q-Block1 and Q-Block2 Option Numbers

This document suggests 19 (TBA1) and 51 (TBA2) as a values to be assigned for the new option numbers.

10.2. New Content Format

This document requests IANA to register the CoAP Content-Format ID for the "application/missing-blocks+cbor-seq" media type in the "CoAP Content-Formats" registry [Format]:

- o Media Type: application/missing-blocks+cbor-seq
- o Encoding: -
- o Id: TBD3
- o Reference: [RFCXXXX]

11. Security Considerations

Security considerations discussed in Section 7 of [RFC7959] should be taken into account.

Security considerations discussed in Sections 11.3 and 11.4 of [RFC7252] should be taken into account. In particular, it is NOT RECOMMENDED that the NoSec security mode is used if the Q-Block1 and Q-Block2 Options are to be used.

Security considerations related to the use of Request-Tag are discussed in Section 5 of [I-D.ietf-core-echo-request-tag].

12. Acknowledgements

Thanks to Achim Kraus, Jim Schaad, and Michael Richardson for the comments on the mailing list.

Special thanks to Christian Amsuess and Carsten Bormann for their suggestions and several reviews, which improved this specification significantly.

Some text from [RFC7959] is reused for readers convenience.

13. References

13.1. Normative References

[I-D.ietf-core-echo-request-tag]
Amsuess, C., Mattsson, J., and G. Selander, "CoAP: Echo, Request-Tag, and Token Processing", draft-ietf-core-echo-request-tag-10 (work in progress), July 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8075] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, DOI 10.17487/RFC8075, February 2017, <https://www.rfc-editor.org/info/rfc8075>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/info/rfc8132>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/rfc8742>.

13.2. Informative References

[Format] , <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>.

[I-D.ietf-dots-telemetry] Boucadair, M., Reddy.K, T., Doron, E., chenmeiling, c., and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry", draft-ietf-dots-telemetry-13 (work in progress), October 2020.

[Options] , <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#option-numbers>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/info/rfc6928>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8782] Reddy.K, T., Ed., Boucadair, M., Ed., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 8782, DOI 10.17487/RFC8782, May 2020, <https://www.rfc-editor.org/info/rfc8782>.

Appendix A. Examples with Confirmable Messages

These examples assume NSTART has been increased to at least 4.

The notations provided in Figure 2 are used in the following subsections.

A.1. Q-Block1 Option

Let's now consider the use Q-Block1 Option with a CON request as shown in Figure 8. All the blocks are acknowledged (ACK).

```
          CoAP        CoAP
         Client      Server
           |           |
           +--------->| CON PUT /path M:0x01 T:0xf0 RT=10 QB1:0/1/1024
           +--------->| CON PUT /path M:0x02 T:0xf1 RT=10 QB1:1/1/1024
           +--------->| CON PUT /path M:0x03 T:0xf2 RT=10 QB1:2/1/1024
           +--------->| CON PUT /path M:0x04 T:0xf3 RT=10 QB1:3/0/1024
           |<---------+ ACK 0.00 M:0x01
           |<---------+ ACK 0.00 M:0x02
           |<---------+ ACK 0.00 M:0x03
           |<---------+ ACK 2.04 M:0x04
```

Figure 8: Example of CON Request with Q-Block1 Option (Without Loss)

Now, suppose that a new body of data is to sent but with some blocks dropped in transmission as illustrated in Figure 9. The client will retry sending blocks for which no ACK was received.

```
          CoAP        CoAP
         Client      Server
           |           |
           +--------->| CON PUT /path M:0x05 T:0xf4 RT=11 QB1:0/1/1024
           +--->X      | CON PUT /path M:0x06 T:0xf5 RT=11 QB1:1/1/1024
           +--->X      | CON PUT /path M:0x07 T:0xf6 RT=11 QB1:2/1/1024
           +--------->| CON PUT /path M:0x08 T:0xf7 RT=11 QB1:3/1/1024
           |<---------+ ACK 0.00 M:0x05
           |<---------+ ACK 0.00 M:0x08
           |           |
         [[The client retries sending packets not acknowledged]]
           +--------->| CON PUT /path M:0x06 T:0xf5 RT=11 QB1:1/1/1024
           +--->X      | CON PUT /path M:0x07 T:0xf6 RT=11 QB1:2/1/1024
           |<---------+ ACK 0.00 M:0x06
           |           |
         [[The client retransmits messages not acknowledged
          (exponential backoff)]]
           +--->?      | CON PUT /path M:0x07 T:0xf6 RT=11 QB1:2/1/1024
           |           |
         [[Either transmission failure (acknowledge retry timeout)
           or successfully transmitted.]]
```

Figure 9: Example of CON Request with Q-Block1 Option (Blocks Recovery)

It is implementation dependent as to whether a CoAP session is terminated following acknowledge retry timeout, or whether the CoAP session continues to be used under such adverse traffic conditions.

If there is likely to be the possibility of network transient losses, then the use of Non-confirmable traffic should be considered.

A.2. Q-Block2 Option

An example of the use of Q-Block2 Option with Confirmable messages is shown in Figure 10.

```
         Client      Server
           |          |
           +--------->| CON GET /path M:0x01 T:0xf0 O:0 QB2:0/0/1024
           |<---------+ ACK 2.05 M:0x01 T:0xf0 O:1234 ET=21 QB2:0/1/1024
           |<---------+ ACK 2.05 M:0xe1 T:0xf0 O:1234 ET=21 QB2:1/1/1024
           |<---------+ ACK 2.05 M:0xe2 T:0xf0 O:1234 ET=21 QB2:2/1/1024
           |<---------+ ACK 2.05 M:0xe3 T:0xf0 O:1234 ET=21 QB2:3/0/1024
               ...
                   [[Observe triggered]]
           |<---------+ CON 2.05 M:0xe4 T:0xf0 O:1235 ET=22 QB2:0/1/1024
           |<---------+ CON 2.05 M:0xe5 T:0xf0 O:1235 ET=22 QB2:1/1/1024
           |<---------+ CON 2.05 M:0xe6 T:0xf0 O:1235 ET=22 QB2:2/1/1024
           |<---------+ CON 2.05 M:0xe7 T:0xf0 O:1235 ET=22 QB2:3/0/1024
           |--------->+ ACK 0.00 M:0xe4
           |--------->+ ACK 0.00 M:0xe5
           |--------->+ ACK 0.00 M:0xe6
           |--------->+ ACK 0.00 M:0xe7
                ...
                   [[Observe triggered]]
           |<---------+ CON 2.05 M:0xe8 T:0xf0 O:1236 ET=23 QB2:0/1/1024
           |     X<---+ CON 2.05 M:0xe9 T:0xf0 O:1236 ET=23 QB2:1/1/1024
           |     X<---+ CON 2.05 M:0xea T:0xf0 O:1236 ET=23 QB2:2/1/1024
           |<---------+ CON 2.05 M:0xeb T:0xf0 O:1236 ET=23 QB2:3/0/1024
           |--------->+ ACK 0.00 M:0xe8
           |--------->+ ACK 0.00 M:0xeb
           |          |
                   [[Server retransmits messages not acknowledged]]
           |<---------+ CON 2.05 M:0xe9 T:0xf0 O:1236 ET=23 QB2:1/1/1024
           |     X<---+ CON 2.05 M:0xea T:0xf0 O:1236 ET=23 QB2:2/1/1024
           |--------->+ ACK 0.00 M:0xe9
           |          |
                   [[Server retransmits messages not acknowledged
                    (exponential backoff)]]
           |     X<---+ CON 2.05 M:0xea T:0xf0 O:1236 ET=23 QB2:2/1/1024
           |          |
             [[Either transmission failure (acknowledge retry timeout)
               or successfully transmitted.]]
```

Figure 10: Example of CON Notifications with Q-Block2 Option

It is implementation-dependent as to whether a CoAP session is terminated following acknowledge retry timeout, or whether the CoAP session continues to be used under such adverse traffic conditions.

If there is likely to be the possibility of network transient losses, then the use of Non-confirmable traffic should be considered.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Jon Shallow
United Kingdom

Email: supjps-ietf@jpshallow.com