CoRE Working Group C. Bormann
Internet-Draft Universitaet Bremen TZI
Intended status: Standards Track June 10, 2016
Expires: December 12, 2016

Block-wise transfers in CoAP: Extension for Reliable Transport (BERT)
draft-bormann-core-block-bert-01

Abstract

CoAP (RFC7252) is a RESTful transfer protocol for constrained nodes and networks, originally using UDP or DTLS over UDP as its transport. Basic CoAP messages work well for the small payloads we expect from temperature sensors, light switches, and similar building-automation devices. CoAP's Block protocol (draft-ietf-core-block) allows transferring larger payloads over limited-size datagrams -- for instance, for firmware updates.

CoAP over TCP and TLS (draft-ietf-core-tcp-tls) enables the use of extended, but not unlimited, size messages. The present specification, Block-wise transfers in CoAP: Extension for Reliable Transport (BERT), extends the block protocol in a simple way to be able to make use of these larger messages over a reliable transport.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

(see abstract for now)

1.1. Objectives

The content of this document is intended for integration into [I-D.ietf-core-coap-tcp-tls].

The objectives stated in the introduction of [I-D.ietf-core-block] apply to the present document as well. (The exception is the desire to enable individual retransmissions -- this is already handled by reliable transport.)

Specifically, this specification continues to minimize the need for creation of additional state, even if a TCP (or TLS over TCP) connection already requires more state than a basic CoAP client-to-server relationship.

An important aspect of this also is the need for state at proxies, see Section 2.1.

1.2. 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 RFC 2119, BCP 14 [RFC2119] and indicate requirement levels for compliant implementations.

The definitions of [RFC7252] apply.

In this document, the term "byte" is used in its now customary sense as a synonym for "octet".

Where bit arithmetic is explained, this document uses the notation familiar from the programming language C, except that the operator "**" stands for exponentiation.

BERT Option:
A Block1 or Block2 option that includes an SZX value of 7.

BERT Block:
The payload of a CoAP message that is affected by a BERT Option in descriptive usage (Section 2.1 of [I-D.ietf-core-block]).

2. BERT Blocks

The use of the present extension is signalled by sending Block1 or Block2 options with SZX == 7 (a "BERT option"). (SZX == 7 is a value that was reserved in [I-D.ietf-core-block].)

In control usage, a BERT option is interpreted in the same way as the equivalent option with SZX == 6, except that it also indicates the capability to process BERT blocks. As with the basic Block protocol, the recipient of a CoAP request with a BERT option in control usage is allowed to respond with a different SZX value, e.g. to send a non-BERT block instead.

In descriptive usage, a BERT option is interpreted in the same way as the equivalent option with SZX == 6, except that the payload is allowed to contain a multiple of 1024 bytes (non-final BERT block) or more than 1024 bytes (final BERT block).

The recipient of a non-final BERT block (M=1) conceptually partitions the payload into a sequence of 1024-byte blocks and acts exactly as if it had received this sequence in conjunction with block numbers

starting at, and sequentially increasing from, the block number given in the Block option. In other words, the entire BERT block is positioned at the byte position that results from multiplying the block number with 1024. The position of further blocks to be transferred is indicated by incrementing the block number by the number of elements in this sequence (i.e., the size of the payload divided by 1024 bytes).

As with SZX == 6, the recipient of a final BERT block (M=0) simply appends the payload at the byte position that is indicated by the block number multiplied with 1024.

2.1. Caching Considerations

Section 2.10 of [I-D.ietf-core-block] applies unchanged.

Discussion: As with the basic Block protocol, a proxy may need to re-slice blocks. Requiring BERT blocks to start at 1024 byte boundaries simplifies this considerably.

2.2. Open Questions

Does the use of CoAP over TCP or TLS simply imply BERT capability or do we explicitly signal that? Signalling is easy for Block2 (but does require sending Block2 options with the value 7 as a matter of course), less so for Block1.

If an optimistic approach is desired, the error code 4.13 (Request Entity Too Large) could be employed as defined in Section 2.5 of [I-D.ietf-core-block].

2.3. Combining BERT blocks with the Observe Option

BERT Blocks combine with the Observe Option ([RFC7641] exactly as defined for basic blocks in Section 2.6 of [I-D.ietf-core-block].

3. Examples

This section extends Section 3 of [I-D.ietf-core-block] with a few examples that involve BERT options. Extending the notation used in that section, a value of SZX == 7 is shown as "BERT", or as "BERT(nnn)" to indicate a payload of size nnn.

In all these examples, a Block option is shown in a decomposed way indicating the kind of Block option (1 or 2) followed by a colon, and then the block number (NUM), more bit (M), and block size exponent (2**(SZX+4)) separated by slashes. E.g., a Block2 Option value of 33

would be shown as 2:2/0/32), or a Block1 Option value of 59 would be shown as 1:3/1/128.

3.1. Block2 Example

The first example (Figure 1) shows a GET request with a response that is split into three BERT blocks. The first response contains 3072 bytes of payload; the second, 5120; and the third, 4711. Note how the block number increments to move the position inside the response body forward.

```
CLIENT                                        SERVER
  |                                             |
  | GET, /status                         ------> |
  |                                             |
  | <------   2.05 Content, 2:0/1/BERT(3072)    |
  |                                             |
  | GET, /status, 2:3/0/BERT             ------> |
  |                                             |
  | <------   2.05 Content, 2:3/1/BERT(5120)    |
  |                                             |
  | GET, /status, 2:8/0/BERT            ------>  |
  |                                             |
  | <------   2.05 Content, 2:8/0/BERT(4711)    |
```

Figure 1: GET with BERT blocks

3.2. Block1 Example

The following example (Figure 2) demonstrates a PUT exchange with BERT blocks.

```
CLIENT                                         SERVER
  |                                              |
  | PUT, /options, 1:0/1/BERT(8192)     ------>  |
  |                                              |
  | <------   2.31 Continue, 1:0/1/BERT          |
  |                                              |
  | PUT, /options, 1:8/1/BERT(16384)    ------>  |
  |                                              |
  | <------   2.31 Continue, 1:8/1/BERT          |
  |                                              |
  | PUT, /options, 1:24/0/BERT(5683)    ------>  |
  |                                              |
  | <------   2.04 Changed, 1:24/0/BERT          |
  |                                              |
```

Figure 2: PUT with BERT blocks

4. IANA Considerations

This specification makes no requests of IANA.

(This section to be removed by the RFC editor.)

5. Security Considerations

The Security Considerations of [I-D.ietf-core-block] apply unchanged.

6. Acknowledgements

7. References

7.1. Normative References

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-20 (work in progress), April 2016.

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

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://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, <http://www.rfc-editor.org/info/rfc7641>.

7.2. Informative References

[I-D.ietf-core-coap-tcp-tls]
Bormann, C., Lemay, S., Technologies, Z., and H. Tschofenig, "A TCP and TLS Transport for the Constrained Application Protocol (CoAP)", draft-ietf-core-coap-tcp-tls-02 (work in progress), April 2016.

Author's Address

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org