CoRE Working Group                                            C. Bormann
Internet-Draft                                 Universitaet Bremen TZI
Intended status: Informational                        October 24, 2010
Expires: April 27, 2011

# Sliced transfers in CoAP
# draft-bormann-core-coap-block-01

Abstract

CoAP is a RESTful transfer protocol for constrained nodes and networks. CoAP is based on datagram transport, which limits the maximum size of resource representations that can be transferred without too much fragmentation. A previous version of this draft defined the Block option. The Block option provides a minimal way to transfer larger representations in a block-wise fashion. It is currently documented in the WG draft draft-ietf-core-block-00.txt

This short I-D attempts to pick up some recent discussion on the CoRE mailing list, and defines a more general, but also slightly more complex approach. It is intended as an example for a complete design based on this discussion, to enable a rational assessment of the relative complexities.

Table of Contents

## 1. Introduction

The CoRE WG is tasked with standardizing an Application Protocol for Constrained Networks/Nodes, CoAP. This protocol is intended to provide RESTful [REST] services not unlike HTTP [RFC2616], while reducing the complexity of implementation as well as the size of packets exchanged in order to make these services useful in a highly constrained network of themselves highly constrained nodes.

This objective requires restraint in a number of sometimes conflicting ways:

- reducing implementation complexity in order to minimize code size,

- reducing message sizes in order to minimize the number of fragments needed for each message (in turn to maximize the probability of delivery of the message), the amount of transmission power needed and the loading of the limited-bandwidth channel,

- reducing requirements on the environment such as stable storage, good sources of randomness or user interaction capabilities.

CoAP is based on datagram transports such as UDP, which limit the maximum size of resource representations that can be transferred without creating unreasonable levels of fragmentation.

A previous version of this draft defined the Block option. The Block option provides a minimal way to transfer larger representations in a block-wise fashion. It is currently documented in a CoAP WG draft [I-D.ietf-core-block].

Recent discussion on the CoRE mailing list centered around the limitations of the Block approach:

- Block is based on fixed block sizes and does not provide for semantic fragmentation of the resource representation.

- The choice of block sizes is limited to a power of two.

- For continuation transfers, the server only receives a numeric block number and a block size, which only enables it to operate in a stateless fashion if the resource representation can be naturally addressed on such block boundaries.

- The negotiation of the desirable message size is quite efficient, but may be simplistic.

This short I-D attempts to pick up this discussion and defines a more general, but also slightly more complex approach. It is intended as an example for a complete design based on this discussion, to enable a rational assessment of the relative complexities.

### 1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP implementations.

The term "byte" is used in its now customary sense as a synonym for "octet".

## 2. Sliced transfers

Not all resource representations will fit into a single link layer packet of a constrained network. Using fragmentation (either at the adaptation layer or at the IP layer) to enable the transport of larger representations is possible up to the maximum size of the underlying datagram protocol (such as UDP), but the fragmentation/reassembly process loads the lower layers with conversation state that is better managed in the application layer.

This specification proposes a set of CoAP options to enable _sliced_ access to resource representations. The overriding objective is to avoid creating conversation state at the server for sliced GET requests. (It is impossible to fully avoid creating conversation state for POST/PUT, if the creation/replacement of resources is to be atomic; where that property is not needed, there is no need to create server conversation state in this case, either.)

Implementation of the options defined in this draft is intended to be optional. However, when one of the Continuation options is present in a CoAP message, it MUST be processed; therefore these options are identified as critical options.

In contrast to the Block option, which provides a limited choice of block sizes and requires all the transfers for one single resource representation retrieval to be of the same size, the options defined in this draft enable a transfer to be structured into a sequence of arbitrarily-sized _slices_, which are of independent size and are only limited by the datagram size of the underlying transport as well as the slice size preferences of the communicating nodes.

### 2.1. The Continuation Options

When a representation is larger than can be comfortably transferred in a single UDP datagram, the Continuation options can be used to indicate a sliced transfer. The value of both Continuation-request and Continuation-response options is an opaque sequence of bytes that is used to link up one slice transaction with the next in a sequence of transactions.

A Continuation-request option in a request indicates that the current transaction is intended as a continuation of the previous transaction that provided the same value for the Continuation-response option in a response.

Where a GET request is sliced up, the Continuation-response option in a response indicates that the slice(s) received so far are not the complete resource representation, but an initial prefix of the

representation. Further bytes of the representation can be retrieved by supplying the value of the Continuation-response option returned in the Continuation-request option of another request that otherwise looks like a GET request.

(TBD: Does the continuation request have to supply the URI and other parameters again along with the Continuation-request option, or does the server make sure the Continuation-response provides all information required, e.g., by sending back a descriptor or by encoding the request parameters in the Continuation-response?)

Each continuation response in a sliced transfer carries the response code that, based on information currently available, the server expects the transfer to receive at the end of the entire sequence of slices, e.g. a 200 code (encoded as 80 decimal) for a successful resource representation retrieval. However, only the response code of the last slice transferred (i.e., without a Continuation-response option) is binding.

### 2.2. Using the Continuation Options

Using the Continuation options, a single REST operation can be split into multiple CoAP message transactions. Each of these message transactions uses their own CoAP transaction ID.

For synchronous responses, the Continuation-request option does not need to be repeated in the response. For an asynchronous response, the server provides both the Continuation-request option and, if applicable, the Continuation-response option.

For GET requests, the server simply adds the Continuation-response option to any response for which it is unable or unwilling to include the entire resource representation. The client then echoes back the value of the Continuation-response in the value of a Continuation-request option. The response carrying the last slice of the representation simply does not contain a Continuation-response option; it is distinguishable from a response carrying the entire resource representation by the presence of Continuation-request in the related request (synchronous) or in the response (asynchronous).

For PUT and POST requests, a client has to indicate that it requests a Continuation-response option from the server whenever it is unable or unwilling to provide the entire (rest of the) representation in the current transaction, i.e. that the slice(s) transferred so far are only a prefix of the entire representation to be transferred. This is done by supplying a Continuation-required option in the request. The request that carries the last slice of the representation does not contain a Continuation-required option.

However, the response to that last forward-transferring request may carry a Continuation-response option, in case the representation returned from the PUT or POST request is larger than can be comfortably transferred in one message. The exchange then continues as it would for a sliced GET request.

### 2.3. Option values for Continuation-request and Continuation-response

The structure of the values provided by the server in a Continuation-response option (and echoed back by the client in a Continuation-request option) is not defined by the protocol. The structure of the value is opaque to the client and entirely determined by the server. In particular, the client is not permitted to manipulate or "make up" Continuation-request values.

For a stateless server, the structure of the value will be chosen to enable generating the next slice. E.g., for a server enumerating devices, the value might be an identifier for the last device described in the last slice.

In order to increase the debuggability of the protocol, there is a suggested structure that can (but need not be) be used whenever it is a good fit. This structure only works when all slices in a transfer (except for the last) use the same slice size, which also is a power of two. If the structure is used, the value of the Continuation-request and Continuation-response options is a sequence of bytes representing an unsigned integer, the three least significant bits of which indicate the size of all slices used in the transfer. The value divided by eight is the number of the slice the Continuation-response pertains to, starting from zero, i.e., the response is about the "size" bytes starting at byte "blocknr << (szx + 3)"; note that the value of the Continuation-request option in the next request echoes the number of the previously transferred block.

```
           0
           0 1 2 3 4 5 6 7
          +-+-+-+-+-+-+-+-+
          | blocknr | szx |
          +-+-+-+-+-+-+-+-+

           0                   1
           0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
          |         block nr        | szx |
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

           0                   1                   2
           0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
          |                 block nr                | szx |
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Suggested structure of the Continuation option values for block-wise transfers, if applicable

### 2.4. The Message-Size option

To influence the slice size used in response to a GET request, the requester uses the (elective) Message-Size option, giving the desired maximum size of a message. A server SHOULD use the slice size indicated or a smaller size.

To influence the slice size used in a PUT or POST request, the server can provide an (elective) Message-Size option in its first response. For the continuation requests, the client SHOULD use the slice size indicated or a smaller size. Since there is no way for the server to influence the size of the first message, the slice carried in this message should be kept reasonably small (e.g., a size that is commensurate with the overhead of the request or even zero bytes in size). A client MAY cache Message-Size options received from a server and use the cached value as a hint for future PUT or POST transactions to that server.

## 3. The Size-Estimate option

(This is new functionality not provided by the Block option.)

The party slicing up a resource representation for sliced transfer may have an idea of the size of the entire resource representation in bytes. Providing this size as an estimate may be beneficial for the other party. If provided, it should be sent with the first slice, and SHOULD provide a close upper bound of the total size that will be transferred. In a PUT or POST transfer, both sides MAY provide a Size-Estimate for their respective resource representation to be transferred.

## 4. IANA Considerations

This draft adds the following option numbers to Table 2 of [I-D.ietf-core-coap]:

| Type | C/E | Name | Data type | Length | Default |
|---|---|---|---|---|---|
| 16 | E | Message-Size | Variable-length unsigned integer | 1-2 B | (none) |
| 17 | C | Continuation-request | Opaque String of Bytes | (any) | (none) |
| 18 | E | Size-Estimate | Variable-length unsigned integer | 1-4 B | (none) |
| 19 | C | Continuation-response | Opaque String of Bytes | (any) | (none) |
| 21 | C | Continuation-required | (none) | 0 | (none) |

## 5. Security Considerations

TBD. (Weigh the security implications of application layer sliced transfer against those of adaptation-layer or IP-layer fragmentation.)

### 5.1. Mitigating Amplification Attacks

TBD. (This section discusses how CoAP nodes could become implicated in DoS attacks by using the amplifying properties of the protocol, as well as mitigations for this threat.)

A CoAP server can reduce the amount of amplification it provides to an attacker by offering large resource representations only in relatively small slices. E.g., for a 1000 byte resource, a 10-byte request might result in an 80-byte response (with a 64-byte block) instead of a 1016-byte response, considerably reducing the amplification provided.

### 5.2. Exposing Server Internals in Continuation-response Values

The server should be careful about the values exposed in Continuation-response options:

- An attacker might be able to derive information from the value that the server did not intend to make available, such as row numbers in an internal database. This can be mitigated by encrypting the value with a secret only known to the server.

- An attacker might attempt to forge Continuation-response values to obtain increased access. Where the structure of the Continuation-response might make this possible, the server should validate it e.g. by including an HMAC computed from a secret only known to the server. If replay attacks might be problem, the server should mitigate them, e.g. by adding a timestamp or a sequence number into the protected data.

## 6. Acknowledgements

As mentioned, much of the content of this draft is the result of discussions within the CoRE mailing list, in particular with Robert Quattlebaum (who insisted on variable-length semantic fragmentation) and Peter Bigot.

# 7. References

## 7.1. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Frank, B., and D. Sturek, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-02 (work in progress), September 2010.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

## 7.2. Informative References

[I-D.ietf-core-block]
Shelby, Z. and C. Bormann, "Blockwise transfers in CoAP", draft-ietf-core-block-00 (work in progress), October 2010.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000.

## Author's Address

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

Phone: +49-421-218-63921
Fax: +49-421-218-7000
Email: cabo@tzi.org