Stream:	Internet Engineering Task Force (IETF)		
RFC:	8710		
Category:	Standards Track		
Published:	February 2020		
ISSN:	2070-1721		
Authors:	T. Fossati	K. Hartke	C. Bormann
	ARM	*Ericsson*	*Universität Bremen TZI*

RFC 8710
Multipart Content-Format for the Constrained Application Protocol (CoAP)

Abstract

This memo defines application/multipart-core, an application-independent media type that can be used to combine representations of zero or more different media types (each with a Constrained Application Protocol (CoAP) Content-Format identifier) into a single representation, with minimal framing overhead.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8710.

Table of Contents

1. Introduction

This memo defines application/multipart-core, an application-independent media type that can be used to combine representations of zero or more different media types (each with a CoAP Content-Format identifier [RFC7252]) into a single representation, with minimal framing overhead.

This simple and efficient binary framing mechanism can be employed to create application-specific message bodies that build on multiple already existing media types.

As the name of the media type suggests, application/multipart-core was inspired by the multipart media types initially defined in the original set of MIME specifications [RFC2046] and later. However, while those needed to focus on the syntactic aspects of integrating multiple representations into one email, transfer protocols providing full data transparency such as CoAP as well as readily available encoding formats such as the Concise Binary Object Representation (CBOR) [RFC7049] shift the focus towards the intended use of the combined representations. In

this respect, the basic intent of the application/multipart-core media type is like that of multipart/mixed (Section 5.1.3 of [RFC2046]); however, the semantics are relaxed to allow for both ordered and unordered collections of media types.

> Historical Note: Experience with multipart/mixed in email has shown that recipients that care about order of included body parts will process them in the order they are listed inside multipart/mixed, and recipients that don't care about the order will ignore it anyway. The media type multipart/parallel that was intended for unordered collections didn't deploy.

The detailed semantics of the representations are refined by the context established by the application in the accompanying request parameters, e.g., the resource URI and any further options (header fields), but three usage scenarios are envisioned:

In one case, the individual representations in an application/multipart-core message body occur in a sequence, which may be employed by an application where such a sequence is natural, e.g., for a number of audio snippets in various formats to be played out in that sequence or search results returned in order of relevance.

In another case, an application may be more interested in a bag of representations (which are distinguished by their Content-Format identifiers), such as an audio snippet and a text representation accompanying it. In such a case, the sequence in which these occur may not be relevant to the application. This specification adds the option of substituting a null value for the representation of an optional part, which indicates that the part is not present.

A third common situation only has a single representation in the sequence, and the sender selects just one of a set of formats possible for this situation. This kind of union "type" of formats may also make the presence of the actual representation optional, the omission of which leads to a zero-length array.

Where these rules are not sufficient, an application might still use the general format defined here but register a new media type and an associated Content-Format identifier to associate the representation with these more specific semantics instead of using the application/multipart-core media type.

Also, future specifications might want to define rough equivalents for other multipart media types with specific semantics not covered by the present specification, such as multipart/alternative (Section 5.1.4 of [RFC2046]), where several alternative representations are provided in the message body, but only one of those is to be selected by the recipient for its use (this is less likely to be useful in a constrained environment that has facilities for pre-flight discovery).

1.1. Requirements Language

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.

2. Multipart Content-Format Encoding

A representation of media type application/multipart-core contains a collection of zero or more representations, each along with their respective Content-Format.

The collection is encoded as a CBOR [RFC7049] array with an even number of elements. Counting from zero, the odd-numbered elements are a byte string containing a representation or the value "null" (if an optional part is indicated as not given). The (even-numbered) element preceding each of these is an unsigned integer specifying the Content-Format ID of the representation following it.

For example, a collection containing two representations, one with Content-Format ID 42 and one with Content-Format ID 0, looks like this in CBOR diagnostic notation:

```
[42, h'0123456789abcdef', 0, h'3031323334']
```

For illustration, the structure of an application/multipart-core representation can be described by the Concise Data Definition Language (CDDL) [RFC8610] specification in Figure 1:

```
multipart-core = [* multipart-part]
multipart-part = (type: uint .size 2, part: bytes / null)
```

Figure 1: CDDL for application/multipart-core

This format is intended as a strict specification: an implementation **MUST** stop processing the representation if there is a CBOR well-formedness error, a deviation from the structure defined above, or any residual data left after processing the CBOR data item. (This generally means the representation is not processed at all unless some streaming processing has already happened.)

3. Usage Example: Observing Resources

This section illustrates a less obvious example for using application/multipart-core: combining it with observing a resource [RFC7641] to handle pending results.

When a client registers to observe a resource for which no representation is available yet, the server may send one or more 2.05 (Content) notifications that indicate the lack of an actual representation. Later on, when one becomes available, the server will send the first actual 2.05 (Content) or 2.03 (Valid) notification. A diagram depicting possible resulting sequences of notifications, identified by their respective response code, is shown in Figure 2.

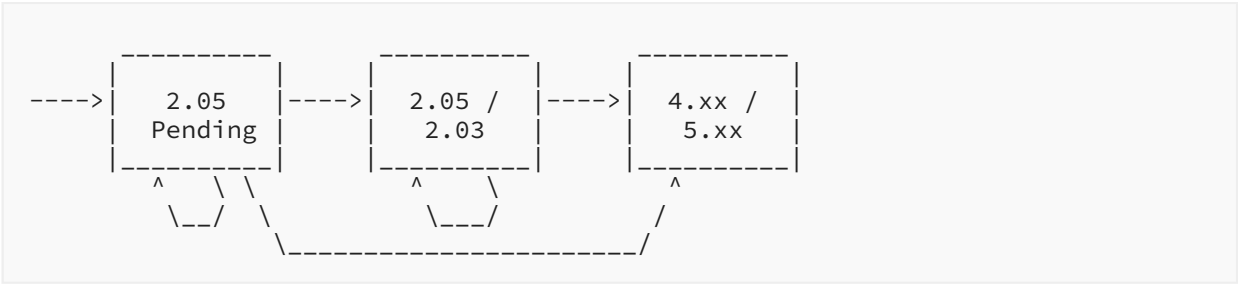

Figure 2: Sequence of Notifications

The specification of the Observe option requires that all notifications carry the same Content-Format. The application/multipart-core media type can be used to provide that Content-Format, e.g., by carrying an empty list of representations in the case marked as "Pending" in Figure 2 and carrying a single representation specified as the target Content-Format in the case in the middle of the figure.

4. Implementation Hints

This section describes the serialization for readers that may be new to CBOR. It does not contain any new information.

An application/multipart-core representation carrying no representations is represented by an empty CBOR array, which is serialized as a single byte with the value 0x80.

An application/multipart-core representation carrying a single representation is represented by a two-element CBOR array, which is serialized as 0x82 followed by the two elements. The first element is an unsigned integer for the Content-Format value, which is represented as described in Table 1. The second element is the object as a byte string, which is represented as a length as described in Table 2 followed by the bytes of the object.

Serialization	Value
0x00..0x17	0..23
0x18 0xnn	24..255
0x19 0xnn 0xnn	256..65535

Table 1: Serialization of Content-Format

Serialization	Length
0x40..0x57	0..23
0x58 0xnn	24..255
0x59 0xnn 0xnn	256..65535

Serialization	Length
0x5a 0xnn 0xnn 0xnn 0xnn	65536..4294967295
0x5b 0xnn .. 0xnn (8 bytes)	4294967296..

Table 2: Serialization of Object Length

For example, a single text/plain object (Content-Format 0) of value "Hello World" (11 characters) would be serialized as follows:

```
0x82 0x00 0x4b H e l l o 0x20 W o r l d
```

In effect, the serialization for a single object is done by prefixing the object with information that there is one object (here: 0x82), information about its Content-Format (here: 0x00), and information regarding its length (here: 0x4b).

For more than one representation included in an application/multipart-core representation, the head of the CBOR array is adjusted (0x84 for two representations, 0x86 for three, etc.), and the sequences of Content-Format and embedded representations follow.

For instance, the example from Section 2 would be serialized as follows:

```
0x84 (*) 0x182A 0x48 0x0123456789ABCDEF (+) 0x00 0x45 0x3031323334
```

where (*) marks the start of the information about the first representation (Content-Format 42, byte string length 8), and (+) marks the start of the second representation (Content-Format 0, byte string length 5).

5. IANA Considerations

5.1. Registration of Media Type application/multipart-core

IANA has registered the following media type [RFC6838]:

Type name: application

Subtype name: multipart-core

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations section of RFC 8710.

Interoperability considerations: N/A

Published specification: RFC 8710

Applications that use this media type: Applications that need to combine representations of zero or more different media types into one, e.g., EST over secure CoAP (EST-CoAP) [EST-COAPS]

Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for application/multipart-core are as specified for application/cbor. (At publication of this document, there is no fragment identification syntax defined for application/cbor.)

Additional information: Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: CoRE WG

Change controller: IESG

Provisional registration? (standards tree only): no

5.2. Registration of a Content-Format Identifier for application/multipart-core

IANA has registered the following Content-Format in the "CoAP Content-Formats" subregistry within the "Constrained RESTful Environments (CoRE) Parameters" registry:

Media Type	Encoding	ID	Reference
application/multipart-core	-	62	RFC 8710

Table 3: Addition to "CoAP Content-Formats" Registry

6. Security Considerations

The security considerations of [RFC7049] apply. In particular, resource exhaustion attacks may employ large values for the byte string size fields or employ deeply nested structures of recursively embedded application/multipart-core representations.

7. References

7.1. Normative References

[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>.

[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>.

7.2. Informative References

[EST-COAPS] Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <https://tools.ietf.org/html/draft-ietf-ace-coap-est-18>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[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>.

[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>.

Acknowledgements

Most of the text in this document is from earlier contributions by two of the authors, Thomas Fossati and Klaus Hartke. This earlier work was reorganized in this document based on the requirements in [EST-COAPS] and discussions with Michael Richardson, Panos Kampanis, and Peter van der Stok.

Authors' Addresses

Thomas Fossati
ARM
Email: thomas.fossati@arm.com

Klaus Hartke
Ericsson
Torshamnsgatan 23
16483 Stockholm
Sweden
Email: klaus.hartke@ericsson.com

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org