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 into a single message, such as a CoAP request or response body, with minimal framing overhead, each along with a CoAP Content-Format identifier.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 22, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must...
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 along with a CoAP Content-Format identifier, into a single representation, with minimal framing overhead. This combined representation may then be carried in a single message, such as a CoAP [RFC7252] request or response body.

This simple and efficient binary framing mechanism can be employed to create application specific request and response bodies which build on multiple already existing media types.

As the name of the media-type suggests, it is inspired by the multipart media types that started to be defined with the original set of MIME specifications [RFC2046]. However, while those needed to focus on the syntactic aspects of integrating multiple representations into one e-mail, 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]). 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:

The individual representations in an application/multipart-core 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 other cases, an application may be more interested in a bag of representations, which are distinguished by their Content-Format identifier, 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 situation that is common only ever has a single representation in the sequence, where the sender already 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 for an application, it 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, 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 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 except if some streaming processing has already happened.)

3. Usage Example: Observing Resources

This section illustrates one 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 before sending 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.
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., 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.

<table>
<thead>
<tr>
<th>Serialization</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00..0x17</td>
<td>0..23</td>
</tr>
<tr>
<td>0x18 0xnn</td>
<td>24..255</td>
</tr>
<tr>
<td>0x19 0xnn 0xnn</td>
<td>256..65535</td>
</tr>
</tbody>
</table>

Table 1: Serialization of content-format
<table>
<thead>
<tr>
<th>Serialization</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40..0x57</td>
<td>0..23</td>
</tr>
<tr>
<td>0x58 0xnn</td>
<td>24..255</td>
</tr>
<tr>
<td>0x59 0xnn 0xnn</td>
<td>256..65535</td>
</tr>
<tr>
<td>0x5a 0xnn 0xnn 0xnn 0xnn</td>
<td>65536..4294967295</td>
</tr>
<tr>
<td>0x5b 0xnn .. 0xnn (8 bytes)</td>
<td>4294967296..</td>
</tr>
</tbody>
</table>

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

0x82 0x00 0x4b Hello World

In effect, the serialization for a single object is done by prefixing the object with information that there is one object (here: 0x82), about its content-format (here: 0x00) and 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, ...) and the sequences of content-format and embedded representations follow.

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

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, (+), of the second representation (content-format 0, byte string length 5).

5. IANA Considerations

5.1. Registration of media type application/multipart-core

IANA is requested to register 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 this

Interoperability considerations: N/A

Published specification: RFC this

Applications that use this media type: Applications that need to combine representations of zero or more different media types into one, e.g., EST-CoAP [I-D.ietf-ace-coap-est]

Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for "application/multipart-core" is 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 is requested to register the following Content-Format to the "CoAP Content-Formats" subregistry, within the "Constrained RESTful
Environments (CoRE) Parameters" registry, from the Expert Review space (0..255):

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Encoding</th>
<th>ID</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/multipart-core</td>
<td>--</td>
<td>TBD1</td>
<td>RFCthis</td>
</tr>
</tbody>
</table>

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 deeply nested structures of recursively embedded application/multipart-core representations.

7. References

7.1. Normative References


7.2. Informative References

Acknowledgements

Most of the text in this draft is from earlier contributions by two of the authors, Thomas Fossati and Klaus Hartke. The re-mix in this document is based on the requirements in [I-D.ietf-ace-coap-est], based on discussions with Michael Richardson, Panos Kampanis and Peter van der Stok.

Authors’ Addresses

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

Klaus Hartke
Ericsson
Torshamngatan 23
Stockholm  SE-16483
Sweden

Email: klaus.hartke@ericsson.com