Abstract

JavaScript Object Notation, JSON (RFC7159) is a text-based data format which is popular for Web based data exchange. Concise Binary Object Representation, CBOR (RFC7049) is a binary data format which has been optimized for data exchange for the Internet of Things (IoT). For many IoT scenarios, CBOR formats will be preferred since it can help decrease transmission payload sizes as well as implementation code sizes compared to other data formats.

Web Linking (RFC5988) provides a way to represent links between Web resources as well as the relations expressed by them and attributes of such a link. In constrained networks, a collection of Web links can be exchanged in the CoRE link format (RFC6690). Outside of constrained environments, it may be useful to represent these collections of Web links in JSON, and similarly, inside constrained environments, in CBOR. This specification defines a common format for this.

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 http://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 January 9, 2017.
1. Introduction

Web Linking [RFC5988] provides a way to represent links between Web resources as well as the relations expressed by them and attributes of such a link. In constrained networks, a collection of Web links can be exchanged in the CoRE link format [RFC6690] to enable resource discovery, for instance by using the CoAP protocol [RFC7252].

The JavaScript Object Notation (JSON) [RFC7159] is a lightweight, text-based, language-independent data interchange format. JSON is popular in the Web development environment as it is easy for humans to read and write.
The Concise Binary Object Representation (CBOR) [RFC7049] is a binary data format which requires extremely small code size, allows very compact message representation, and provides extensibility without the need for version negotiation. CBOR is especially well suited for IoT environments because of these efficiencies.

When converting between a bespoke syntax such as that defined by [RFC6690] and JSON or CBOR, many small decisions have to be made. If left without guidance, it is likely that a number of slightly incompatible dialects will emerge. This specification defines a common approach for translating between the CoRE-specific bespoke formats, JSON and CBOR formats. Where applicable, mapping from other formats (e.g. CoRE Link Format) into JSON or CBOR is also described.

This specification defines a common format for representing CoRE Web Linking in JSON and CBOR.

Note that there is a separate question on how to represent Web links pointing out of JSON documents, as discussed e.g. in [MNOT11]. While there are good reasons to stay as compatible as possible to developments in this area, the present specification is solving a different problem.

1.1. Objectives

This specification has been designed based on the following objectives:

- Canonical mapping
  - lossless round-tripping with [RFC6690] and between JSON and CBOR
  - but not trying for bit-preserving (DER-style) round-tripping

- The simplest thing that could possibly work
  - Do not cater for RFC 5988 complications caused by HTTP header character set issues [RFC2047]

- Consider other work that has links in JSON, e.g.: JSON-LD, JSON-Reference [I-D.pbryan-zyp-json-ref]
  - Do not introduce unmotivated differences
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 [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

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

CoAP: Constrained Application Protocol [RFC7252]

CBOR: Concise Binary Object Representation [RFC7049]

CoRE: Constrained RESTful Environments, the field of work underlying [RFC6690], [RFC7049], [RFC7252], and [RFC7641]

IoT: Internet of Things

JSON: JavaScript Object Notation [RFC7159]

The objective of the JSON and CBOR mappings defined in this document is to contain information of the formats specified in [RFC5988] and [RFC6690]. This specification therefore uses the names of the ABNF productions used in those documents.

2. Web Links in JSON and CBOR

2.1. Background

Web Linking [RFC5988] provides a way to represent links between Web resources as well as the relations expressed by them and attributes of such a link. In constrained networks, a collection of Web links can be exchanged in the CoRE link format [RFC6690] to enable resource discovery, for instance by using the CoAP protocol [RFC7252] and in conjunction with the CoRE resource directory [I-D.ietf-core-resource-directory].

2.2. Information Model

This section discusses the information model underlying the CORE Link Format payload.

An application/link-format document is a collection of web links ("link-value"), each of which is a collection of attributes ("link-param") applied to a "URI-Reference".
We straightforwardly map:

- the outer collection to an array of links;
- each link to a JSON object or CBOR map, mapping attribute names to attribute values.

In the object representing a "link-value", each target attribute or other parameter ("link-param") is represented by a JSON name/value pair (member). The name is a string representation of the parameter or attribute name (as in "parmname"), the value is a string representation of the parameter or attribute value ("ptoken" or "quoted-string"). "quoted-string" productions are parsed (i.e, the outer quotes removed and the backslash constructions evaluated) as defined in [RFC6690] and its referenced documents, before placing them in JSON strings (in the representation of which they may gain back additional decorations such as backslashes as defined in [RFC7159]).

If no attribute value ("ptoken" or "quoted-string") is present, the presence of the attribute name is indicated by using the Boolean value "true" as the value.

If a Link attribute ("parmname") is present more than once in a "link-value", its values are then represented as a JSON array of JSON string values; this array becomes the value of the JSON name/value pair where the attribute name is the JSON name. Attributes occurring just once MUST NOT be represented as JSON arrays but MUST be directly represented as JSON strings. (Note that [RFC6690] has cut down on the use of repeated parameter names; they are still allowed by [RFC5988] though. No attempt has been made to decode the possibly space-separated values for rt=, if=, and rel= into JSON arrays.)

The URI-Reference is represented as a name/value pair with the name "href" and the URI-Reference as the value. (Rationale: This usage is consistent with the use of "href" as a query parameter for link-format query filtering and with link-format reserving the link parameter "href" specifically for this use [RFC6690]).

The resulting structure can be represented in CDDL [I-D.greevenbosch-appsawg-cbor-cddl] as:
links = [* link]
link = {
    href: tstr ; resource URI
    * tstr => tstr / true
}

Figure 1: CoRE Link Format Data Model

2.3.  Additional Encoding Step for CBOR

The above specification for JSON could be used as is for the CBOR encoding as well. However, to further reduce message sizes, an extra encoding step is performed: "href" and some commonly occurring attribute names are encoded as small integers.

The substitution is summarized below:

<table>
<thead>
<tr>
<th>name</th>
<th>encoded value</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>href</td>
<td>1</td>
<td>[RFC6690], [RFCthis]</td>
</tr>
<tr>
<td>rel</td>
<td>2</td>
<td>[RFC5988] Section 5.3</td>
</tr>
<tr>
<td>anchor</td>
<td>3</td>
<td>[RFC5988] Section 5.2</td>
</tr>
<tr>
<td>rev</td>
<td>4</td>
<td>[RFC5988] Section 5.3</td>
</tr>
<tr>
<td>hreflang</td>
<td>5</td>
<td>[RFC5988] Section 5.3</td>
</tr>
<tr>
<td>media</td>
<td>6</td>
<td>[RFC5988] Section 5.4</td>
</tr>
<tr>
<td>title</td>
<td>7</td>
<td>[RFC5988] Section 5.4</td>
</tr>
<tr>
<td>type</td>
<td>8</td>
<td>[RFC5988] Section 5.4</td>
</tr>
<tr>
<td>rt</td>
<td>9</td>
<td>[RFC6690] Section 3.1</td>
</tr>
<tr>
<td>if</td>
<td>10</td>
<td>[RFC6690] Section 3.2</td>
</tr>
<tr>
<td>sz</td>
<td>11</td>
<td>[RFC6690] Section 3.3</td>
</tr>
<tr>
<td>ct</td>
<td>12</td>
<td>[RFC7252] Section 7.2.1</td>
</tr>
<tr>
<td>obs</td>
<td>13</td>
<td>[RFC7641] Section 6</td>
</tr>
<tr>
<td>ins</td>
<td>14</td>
<td>[I-D.ietf-core-resource-directory]</td>
</tr>
<tr>
<td>exp</td>
<td>15</td>
<td>[I-D.ietf-core-resource-directory]</td>
</tr>
</tbody>
</table>

Table 1: Integer Encoding of common attribute names

(Comment to be deleted before submitting this document to the IESG: This list should, again, be checked against relevant references at WGLC time.)

This list of substitutions is fixed by the present specification; no future expansion of the list is foreseen. "href" as well as all attribute names in this list MUST be represented by their integer substitutions and MUST NOT use the attribute name in text form.
This leads to the following CDDL representation for the CBOR encoding:

```
links = [* link]
link = {
    href: tstr ; resource URI
    * label => tstr / true
}
label = tstr / &(
    href: 1, rel: 2, anchor: 3,
    rev: 4, hreflang: 5, media: 6,
    title: 7, type: 8, rt: 9,
    if: 10, sz: 11, ct: 12,
    obs: 13,
)
```

Figure 2: CoRE Link Format Data Model (CBOR)

### 2.4. Examples

The examples in this section are based on an example on page 15 of [RFC6690] (Figure 3).

```
</sensors>;ct=40;title="Sensor Index",
</sensors/temp>;rt="temperature-c";if="sensor",
</sensors/light>;rt="light-lux";if="sensor",
<http://www.example.com/sensors/t123>;anchor="/sensors/temp";
rel="describedby",
</t>;anchor="/sensors/temp";rel="alternate"
```

Figure 3: Example from page 15 of [RFC6690]

#### 2.4.1. Link Format to JSON Example

The link-format document in Figure 3 becomes (321 bytes, line breaks shown are not part of the minimally-sized JSON document):

```
"["href":"/sensors","ct":"40","title":"Sensor Index"],
["href":"/sensors/temp","rt":"temperature-c","if":"sensor"],
["href":"/sensors/light","rt":"light-lux","if":"sensor"],
["href":"http://www.example.com/sensors/t123","anchor":"/sensors/temp","rel":"describedby"],
["href":"/t","anchor":"/sensors/temp","rel":"alternate"]"
```

To demonstrate the handling of value-less and array-valued attributes, we extend the link-format example by examples of these
(Figure 4; the "obs" attribute is defined in Section 6 of [RFC7641], while the "foo" attribute is for exposition only):

```xml
<sensors>;ct=40;title="Sensor Index",
</sensors/temp>;rt="temperature-c";if="sensor";obs,
</sensors/light>;rt="light-lux";if="sensor",
<http://www.example.com/sensors/t123>;anchor="/sensors/temp";
rel="describedby";foo="bar";foo=3;ct=4711,
</t>;anchor="/sensors/temp";rel="alternate"
```

Figure 4: Example derived from page 15 of [RFC6690]

The link-format document in Figure 4 becomes the JSON document in Figure 5 (some spacing and indentation added):

```json
[{
"href": "/sensors",
"ct": "40",
"title": "Sensor Index"
},
{
"href": "/sensors/temp",
"rt": "temperature-c",
"if": "sensor",
"obs": true
},
{
"href": "/sensors/light",
"rt": "light-lux",
"if": "sensor"
},
{
"href": "http://www.example.com/sensors/t123",
"anchor": "/sensors/temp",
"rel": "describedby",
"foo": ["bar", "3"],
"ct": "4711"
},
{
"href": "/t",
"anchor": "/sensors/temp",
"rel": "alternate"
}]
```

Figure 5: Example derived from page 15 of [RFC6690]

Note that the conversion is unable to convert the string-valued "ct" attribute to a number, which would be the natural type for a Content-Format value; similarly, both "foo" values are treated as strings independently of whether they are quoted or numeric in syntax.

2.4.2. Link Format to CBOR Example

This examples shows conversion from link format to CBOR format.

The link-format document in Figure 3 becomes (in CBOR diagnostic format):

```cbor
[1: "/sensors", 12: "40", 7: "Sensor Index"],
 2: "describedby"],
```

or, in hexadecimal (203 bytes):

```
85 # array(number of data items:5)
```
a3   # map(# data item pairs:3)
01   # unsigned integer(value:1, "href")
68   # text string(8 bytes)
     # "/sensors"
2f73656e736f7273       # "/sensors"
0c   # unsigned integer(value:12, "ct")
62   # text(2)
     # "40"
3430  # unsigned integer(value:7, "title")
07   # text string(12 bytes)
     # "Sensor Index"
6c   # text string(35 bytes)
53656e736f7220496e646578 # "Sensor Index"
     # text string(35 bytes)
     # "http://www.example.com/sensors/t123"
03   # unsigned integer(value:3, "anchor")
6d   # text string(13 bytes)
2f73656e736f72732f74   # "/sensors/light"
696676742d6c7578       # "light-lux"
0a   # unsigned integer(value:10, "if")
66   # text string(6 bytes)
     # "sensor"
73656e736f72       # text string(35 bytes)
     # "/sensors/light"
09   # unsigned integer(value:9, "rt")
69   # text string(9 bytes)
6c696768742d6c7578   # "light-lux"
0a   # unsigned integer(value:10, "if")
66   # text string(6 bytes)
     # "sensor"
73656e736f72       # text string(35 bytes)
     # "http://www.example.com/sensors/t123"
83   # unsigned integer(value:3, "anchor")
23   # text string(13 bytes)
687474703a2f2f777777   # "http://www.example.com/sensors/t123"
2e6578616d706c652e63   # http://www.example.com/sensors/t123
6f6d2f73656e736f7273   # http://www.example.com/sensors/t123
2f74313233   # "http://www.example.com/sensors/t123"
03   # unsigned integer(value:3, "anchor")
6d   # text string(13 bytes)
2f73656e736f72732f74   # "/sensors/temp"
696676742d6c7578       # "light-lux"
0a   # unsigned integer(value:10, "if")
66   # text string(6 bytes)
     # "describedby"
73656e736f72       # text string(35 bytes)
83   # unsigned integer(value:3, "anchor")
23   # text string(13 bytes)
687474703a2f2f777777   # "http://www.example.com/sensors/t123"
2e6578616d706c652e63   # http://www.example.com/sensors/t123
6f6d2f73656e736f7273   # http://www.example.com/sensors/t123
2f74313233   # "http://www.example.com/sensors/t123"
03   # unsigned integer(value:3, "anchor")
6d   # text string(13 bytes)
2f73656e736f72732f74   # "/sensors/temp"
696676742d6c7578       # "light-lux"
0a   # unsigned integer(value:10, "if")
66   # text string(6 bytes)
     # "describedby"
73656e736f72       # text string(35 bytes)
83   # unsigned integer(value:3, "anchor")
23   # text string(13 bytes)
687474703a2f2f777777   # "http://www.example.com/sensors/t123"
2e6578616d706c652e63   # http://www.example.com/sensors/t123
6f6d2f73656e736f7273   # http://www.example.com/sensors/t123
2f74313233   # "http://www.example.com/sensors/t123"
3. IANA Considerations

This specification registers the following additional Internet Media Types:
Type name: application

Subtype name: link-format+json

Required parameters: None

Optional parameters: None

Encoding considerations: Resources that use the "application/ link-format+json" media type are required to conform to the "application/json" Media Type and are therefore subject to the same encoding considerations specified in [RFC7159], Section 11.

Security considerations: As defined in this specification

Published specification: This specification.

Applications that use this media type: None currently known.

Additional information:

  Magic number(s): N/A

  File extension(s): N/A

  Macintosh file type code(s): TEXT

Person & email address to contact for further information:
Carsten Bormann <cabo@tzi.org>

Intended usage: COMMON

Change controller: IESG

and
Type name: application
Subtype name: link-format+cbor
Required parameters: None
Optional parameters: None

Encoding considerations: Resources that use the "application/link-format+cbor" media type are required to conform to the "application/cbor" Media Type and are therefore subject to the same encoding considerations specified in [RFC7049], Section 7.

Security considerations: As defined in this specification

Published specification: This specification.

Applications that use this media type: None currently known.

Additional information:
  Magic number(s): N/A
  File extension(s): N/A
  Macintosh file type code(s): CBOR

Person & email address to contact for further information:
Kepeng Li <kepeng.lkp@alibaba-inc.com>

Intended usage: COMMON
Change controller: IESG

4. Security Considerations

The security considerations relevant to the data model of [RFC6690], as well as those of [RFC7049] and [RFC7159] apply.

5. References

5.1. Normative References


5.2. Informative References

[I-D.greevenbosch-appsawg-cbor-cddl]

[I-D.ietf-core-resource-directory]

[I-D.pbryan-zyp-json-ref]


A reference implementation of a converter from RFC6690 link-format to JSON and CBOR (and back to link-format) in the programming language Ruby [RUBY] is reproduced below.

```ruby
require 'strscan'
require 'json'
require 'cbor-pretty'
class String
  def as_utf8
    force_encoding(Encoding::UTF_8)
  end
end
module CoRE
  module Links
    def self.map_to_true(a)
      Hash[a.map{ |t| [t, true]}}
    end
    PTOKENCHAR = %r"[\[\]\w!#-+-/:<-?^-'{}~@]"
    QUOSTRCHAR = %r{(?:\[^"\]\|\.)}    # to be used inside "
    ATTRCHAR   = %r"\[\w!#$&+.^'|~-\]
    MUSTBEQUOTED = map_to_true(%w{anchor title rt if})
    ANCHORNAME = "href"
    SCANATTR = %r{(#\w)(\w[\w\d]*)(\w[\w\d]*)\".*)?} # "
    RAWMAPPINGS = <<-DATA
      href: 1, rel: 2, anchor: 3,
      rev: 4, hreflang: 5, media: 6,
      title: 7, type: 8, rt: 9,
      if: 10, sz: 11, ct: 12,
      obs: 13,
      DATA
    MAPPPINGS = Hash.new { |h, k| k}
```
RAWMAPPINGS.scan(/([-\w]+)\s*:\s*([-\w]+),/) do |n, v|
    MAPPINGS[n] = Integer(v)
end

def self.parse(*args)
    WLNK.parse(*args)
end

class WLNK
    attr_accessor :resources
    def initialize(r = [])
        # make sure the keys are strings
        @resources = r.to_ary
        # make sure it’s an Array
    end
    def self.parse(s, robust = true)
        wl = WLNK.new
        ss = StringScanner.new(s.as_utf8)
        ss.skip(/\s+/) if robust
        while ss.scan(%r{<\[^>]+>})
            res = { ANCHORNAME => ss[1].as_utf8 }
            ss.skip(/\s*/+) if robust
            while ss.skip(/;/)
                ss.skip(/\s*/+) if robust
                unless ss.scan(SCANATTR)
                    raise ArgumentError, "must have attribute behind ";'
                    at: #{ss.peek(20).inspect} (byte #{ss.pos})"
                end
                key = ss[1].as_utf8
                value = ss[2] ||
                if res[key]
                    res[key] = Array(res[key]) << value
                else
                    res[key] = value
                end
                ss.skip(/\s*/+) if robust
            end
            wl.resources << res
            break unless ss.skip(/,/)  
            ss.skip(/\s*/+) if robust
        end
        ss.skip(/\s*/+) if robust
        raise ArgumentError, "link-format unparseable at:
        #{ss.peek(20).inspect} (byte #{ss.pos})" unless ss.eos?
        wl
    end
    def to_json
        JSON.pretty_generate(@resources)
    end
end
def to_cbor
    CBOR.encode(@resources.map { |r|
        Hash[r.map { |k, v| [MAPPINGS[k], v] }])
    end

def to_wlnk
    resources.map do |res|
        res = res.dup
        u = res.delete(ANCHORNAME)
        "<#{u}>", *res.map do |k, v|
            if String === v
                if MUSTBEQUOTED[k] || v =~ /\A#{PTOKENCHAR}+\z/ # RFC 6690
                    "#{k}="%{v.gsub(/\[\"\]/) { |x| "\"\#{x}\""}}"
                else
                    "#{k}=#{v}"
                end
            else
                "#{k}" end
        end].join(‘;’)
        end.join("","")
    end
end

lf = CoRE::Links.parse(ARGF.read)

puts lf.to_json # JSON
puts CBOR.pretty(lf.to_cbor) # CBOR "pretty" binary form
puts lf.to_wlnk # RFC 6690 link-format
# <CODE ENDS>

Acknowledgements

Special thanks to Bert Greevenbosch who was an author on the initial
version of a contributing document as well as the original author on
the CDDL notation.

Hannes Tschofenig made many helpful suggestions for improving this
document.
Authors’ Addresses

Kepeng LI
Alibaba Group
Wenxi Road, Yuhang District
Hangzhou, Zhejiang 311121
China
Email: kepeng.lkp@alibaba-inc.com

Akbar Rahman
InterDigital Communications, LLC
1000 Sherbrooke Street West
Montreal, Quebec H3A 3G4
Canada
Phone: +1-514-585-0761
Email: akbar.rahman@interdigital.com

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