Thing-to-Thing Research Group
Internet-Draft
Intended status: Experimental
Expires: September 14, 2016

K. Hartke
Universitaet Bremen TZI

March 13, 2016

The Constrained RESTful Application Language (CoRAL)
draft-hartke-t2trg-coral-00

Abstract

The Constrained RESTful Application Language (CoRAL) is a compact, binary representation format for hypermedia-driven applications. It supports links, forms and the embedding of resource representations.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Constrained RESTful Environments (CoRE) realize the Web architecture [W3C.REC-webarch-20041215] in a suitable form for constrained nodes and networks [RFC7228].

In the Web, hypertext documents contain links and forms that allow a user to navigate between resources and submit data to a server for processing. By annotating these elements with machine-readable link relation types [RFC5988] and form relation types, it is possible to extend this interaction model to machine-to-machine communication.

This document describes the Constrained RESTful Application Language (CoRAL), a serialization format for Web links and forms that is based on the Concise Binary Object Representation (CBOR) [RFC7049]. The format also supports the embedding of representations of resources. Thus, CoRAL can be used as a building block for RESTful, hypermedia-driven applications.

1.1. Terminology

Readers are expected to be familiar with the terms and concepts described in [RFC5988] and [I-D.hartke-core-apps].

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].

2. Overview

2.1. Links

A Web link [RFC5988] is a typed connection between two Web resources. It is comprised of a context, a link relation type, a link target URI and, optionally, a set of target attributes. The link relation type identifies the semantics of a link and thus enables an automated agent to understand the nature of the relation.

The CoRAL representation of a resource contains a set of links where the context of each link is the represented resource. A link in a CoRAL representation serializes the link relation type, the URI of the link target and the target attributes, if any.

In the Web, link relation types are identified by strings, such as "stylesheet", "terms-of-service" or "item". In order to minimize the overhead of using these relation types in constrained environments, [I-D.hartke-core-apps] extends the link relation types registry with a numeric identifier for each link relation type. CoRAL uses this numeric identifier instead of the link relation type name.

The link target URI and the target attributes are encoded as options in a simple format based on the option structure in CoAP [RFC7252] and CBOR encoding [RFC7049]. For example, the Web link (in [RFC5988] syntax)

```
<coap://example.com/info/tos>;rel=terms-of-service;type=text/plain
```

is serialized in CoRAL as follows:

```
      [ /abs_link/          0,
        /terms-of-service/ 64,
        [ /format/          3, 0 /text//plain/,
          /href.scheme/     4, "coap",
          /href.host.name/  6, "example.com",
          /href.port/      11, 5683,
          /href.path/      12, "info",
          /href.path/      12, "tos" ]]
```

Multiple links are simply serialized one after another:

```
      [ /rel_link/     4,
        /first/       23,
        [ /format/     3, 0 /text//plain/,
          /href.path/ 12, "page1" ]]
      [ /rel_link/     4,
        /previous/    49,
        [ /format/     3, 0 /text//plain/,
          /href.path/ 12, "page6" ]]
      [ /rel_link/     4,
        /next/        38,
        [ /format/     3, 0 /text//plain/,
          /href.path/ 12, "page8" ]]
      [ /rel_link/     4,
        /last/        31,
        [ /format/     3, 0 /text//plain/,
          /href.path/ 12, "page42" ]]
```

The Format Option, when present, is a hint indicating what the CoAP content format of the result of dereferencing the link should be. If more than one format is available, the Format Option can be repeated:

```
      [ /rel_link/     4,
        /item/        30,
        [ /format/     3, 47 /application//exi/,
          /format/     3, 50 /application//json/,
          /format/     3, 60 /application//cbor/,
          /href.path/ 12, "item1" ]]
```

2.2. **Embedding**

If a representation links to many resources, it may be inefficient to retrieve a representation of each link target individually. For this reason, CoRAL supports the embedding of a representation of the link target in the link itself:

```
      [ /rel_link/     4,
        /item/        30,
        [ /format/     3, 50 /application//json/,
          /href.path/ 12, "item1" ],
        <<<
          {
            "task":     "Return the books to the library",
            "assignee": "Alice"
          }
        >>> ]
```

By embedding representations, it is possible to use CoRAL as a (very basic) substitute for RDF [W3C.REC-rdf11-concepts-20140225]. For example, the RDF graph (in Turtle [W3C.REC-turtle-20140225] syntax)

```
      @prefix foaf: <http://xmlns.com/foaf/0.1/> .

      <> foaf:name     "John Doe" ;
         foaf:age      32 ;
         foaf:homepage <coap://www.doe.example/> .
```

could be serialized in CoRAL as follows:

```
      [ /anon_link/          12,
        /name/             -100,
        [ /format/            3, 0 /text//plain/ ],
        <<<
          John Doe
        >>> ]
      [ /anon_link/          12,
        /age/              -101,
        [ /format/            3, 9 /uint8/ ],
        <<<
          32
        >>> ]
      [ /abs_link/            0,
        /homepage/         -102,
        [ /href.scheme/       4, "coap",
          /href.host.name/    6, "www.doe.example" ]]
```

A flag in the serialized link indicates that the targets of the first two links are anonymous resources that don't have their own URI, like literals in RDF.

2.3. Namespaces

The link relation type in a serialized link may be from the "global" or the "local" namespace. The global namespace is indicated by an unsigned number and is made up of the link relation types registered with IANA. The local namespace is indicated by a negative number and is defined by the media type of the CoRAL representation.

By default, CoRAL representations have the "application/coral" media type where the local namespace is empty. However, it is possible to create new media types based on CoRAL and to register these with the "+coral" suffix. In this case, the media type specification can fill the local namespace with application-specific link relation types.

For example, a media type "application/example.shop+coral" could define the following set of local link relation types:

ID	Meaning
80	http://example.com/rels/order
81	http://example.com/rels/basket
82	http://example.com/rels/customer

Similarly, a media type "application/example.foaf+coral" could define the following mapping from local link relation type IDs to the FOAF RDF model [FOAF]:

ID	Meaning
100	http://xmlns.com/foaf/0.1/name
101	http://xmlns.com/foaf/0.1/age
102	http://xmlns.com/foaf/0.1/homepage

2.4. Forms

In addition to Web links, CoRAL also supports forms. A form is an affordance that an agent can use to perform an operation on the form context, such as updating a resource or creating a new item in a collection.

In a form, the link relation type is replaced by the form relation type which indicates the semantics of the form. The Href.* Options encode the URI of the target resource to which the agent should submit the form. A form additionally encodes the submission method

(POST, PUT, PATCH, DELETE) and the description of a representation that the service accepts as part of form submission:

```
   [ /rel_form/    68,
     /create-item/  1,
     [ /method/     1, 2  /POST/,
       /accept/     2, 60 /application//cbor/,
       /href.path/ 12, "items" ]]
```

The Accept Option specifies the content format of the accepted representation. A content format may use the payload of a form to describe the accepted representation in more detail, for example, by specifying a set of form fields that the agent needs to fill out:

```
   [ /rel_form/    68,
     /create-item/  1,
     [ /method/     1, 2     /POST/,
       /accept/     2, 65535 /example//form/,
       /href.path/ 12, "items" ],
     <<<
       name, age, homepage
     >>> ]
```

2.5. Editing

The target resource of a link may be editable. In this case, the representation of such a resource typically contains a form that allows to edit the resource. However, it may be inefficient to include this form in every representation of the target resource and more efficient to include it in a representation that links to the resource. CoRAL supports this by two flags in the link structure.

Setting the Updateable Flag in a link defines a form that can be used to update the target resource. The context and target resource of that form is the target resource of the link, the submission method is PUT and the content format of the submitted representation is defined by the Format Option of the link. For example, given the following CoRAL representation an agent can change the recipient by making a PUT request to <./to> with the new value in text/plain format:

```
      [ /updateable_rel_link/    6,
        /sender/              -120,
        [ /format/               3, 0 /text//plain/,
          /href.path/           12, "from" ],
        <<<
          Juliet
        >>> ]
      [ /updateable_rel_link/    6,
        /recipient/           -121,
        [ /format/               3, 0 /text//plain/,
          /href.path/           12, "to" ],
        <<<
          Romeo
        >>> ]
      [ /updateable_rel_link/    6,
        /message/             -122,
        [ /format/               3, 0 /text//plain/,
          /href.path/           12, "message" ],
        <<<
          Art thou not Romeo, and a Montague?
        >>> ]
```

Setting the Deleteable Flag in a link likewise defines a form that can be used to delete the target resource. The submission method is DELETE and no representation is included in the request.

3. Data Format

A CoRAL representation consists of a sequence of zero or more links and/or forms, each encoded in CBOR [RFC7049] and concatenated as a byte string. Before encoding, both links and forms are arrays with either three of four elements: an unsigned integer carrying flags, an integer for the link or form relation type, an array that contains zero or more options (as pairs of option numbers and option values, ordered by option number), and optionally a payload.

Using the notation of [I-D.greevenbosch-appsawg-cbor-cddl], this can be expressed as:

```
      link_or_form = [flags, relation, options, ? payload]
      flags = uint .bits flagbits
      relation = int          ; negative for local
      options = [* (optionname, optionvalue)]
      optionname = uint
      optionvalue = uint / text / bytes
      payload = bytes
      flagbits = &(
        deleteable: 0,
        updateable: 1,
        hreftype1: 2,
        hreftype2: 3,
        is_link: 6,
      )
```

```
                             +-+-+-+-+-+-+-+-+
                             |_|T|_|_| H |U|D|
                             +-+-+-+-+-+-+-+-+
```

Figure 1: Flags

The flags unsigned integer contains the following fields:

Type (T): A 1-bit unsigned integer. Indicates whether the structure is a link (0) or a form (1).

Href Type (H): A 2-bit unsigned integer. Indicates whether the link target is specified as an absolute URI (0), as a relative URI (1) or as the default URI (2), or whether the target is an anonymous resource (3).

Updateable (U): A 1-bit unsigned integer. Indicates whether the link target can be updated (1) or not (0).

Deleteable (D): A 1-bit unsigned integer. Indicates whether the link target can be deleted (1) or not (0).

The relation type is an integer that indicates the link or form relation type. Negative integers are used for local relation types, unsigned ones for global relation types.

The options are an array that contains a sequence of pairs of a CoRAL option number and an option value, where the option numbers are in ascending order.

The payload is an optional element of the array.

3.1. Option Format

CoRAL defines a number of options that can be included in links and forms. Options are used to encode the target resource URI and the target attributes. An option instance in a link or form maps the option number of a defined CoRAL option to the option value.

4. Options

The CoRAL options defined in this document are summarized in Table 1 below and explained in the following subsections:

No.	R	Name	Format	Length	Default
0		Id	string	1-255	(none)
1		Method	uint	1	2 (POST)
2	x	Accept	uint	0-2	(none)
3	x	Format	uint	0-2	(none)
4		Href.Scheme	string	1-255	(none)
6		Href.Host.Name	string	1-255	(none)
7		Href.Host.IPv4	opaque	4	(none)
8		Href.Host.IPv6	opaque	16	(none)
11		Href.Port	uint	0-2	(see below)
12	x	Href.Path	string	0-255	(none)
13	x	Href.Query	string	0-255	(none)
14		Href.Fragment	string	0-255	(empty)
15		Other-Href	string	1-1034	(none)
20		Title	string	0-255	(none)

Table 1: Options

The option properties are defined as follows:

Number: An option is identified by an option number.

Repeatable (R): An option that is repeatable MAY be included one or more times in a link or form. An option that is not repeatable MUST NOT be included more than once. If an agent encounters an option with more occurrences than the option is defined for, each supernumerary option occurrence MUST be ignored.

Format: Option values are defined to have a certain format. Similar to the types defined in Section 3.2 of RFC 7252, "string" stands for a text string; "opaque" for a byte string, and "uint" for an unsigned integer.

Length: Option values are defined to have a specific length, often in the form of an upper and lower bound. For unsigned integer options the length is counted as the number of bytes that would be needed to represent the unsigned integer as a binary number. The length of an option value MUST NOT be outside the defined range. If an agent encounters an option with a length outside the defined range, that option MUST be ignored.

Default Value: Options may be defined to have a default value. If the value of an option is intended to be this default value, the option SHOULD NOT be included in the link or form. If the option is not present, the default value MUST be assumed.

4.1. Accept

The Accept Option indicates the acceptable content formats for the representation included in a form submission. Each option value is one of the content format IDs defined in the CoAP Content-Formats registry. If the Accept Option is absent, the service accepts any content format.

4.2. Format

The Format Option, when present in a link or a form, is a hint indicating what the content format of the payload of the CoAP response should be when following the link or submitting the form. Note that this is only a hint; it does not override the Content-Format Option included in the CoAP response.

As an exception to this rule, the Format Option is REQUIRED if a link embeds a representation. The option then indicates the CoAP content format of the embedded representation.

Each option value is one of the content format IDs defined in the CoAP Content-Formats registry.

4.3. Href.*

The Href.Scheme, Href.Host.Name, Href.Host.IPv4, Href.IPv6, Href.Port, Href.Path, Href.Query and Href.Fragment Options are used to specify the target resource URI in a link or form. They hold the following values:

o the Href.Scheme Option specifies the URI scheme name,

o the Href.Host.Name Option specifies the host as a registered name [RFC3986],

- o the Href.Host.IPv4 Option specifies the host as a 32-bit IPv4 address,
- o the Href.Host.IPv6 Option specifies the host as a 128-bit IPv6 address,
- o the Href.Port Option specifies the port number,
- o each Href.Path Option specifies one segment of the path,
- o each Href.Query Option specifies one argument of the query string, and
- o the Href.Fragment Option specifies the fragment identifier.

The Href.Host.Name, Href.Host.IPv4 and Href.Host.IPv6 options are mutually exclusive.

The default value of the Href.Port Option is the default port for the URI scheme.

The following table lists the permitted Href.* options by Href Type. A 'yes' indicates that an option of this type MAY be present; a 'no' indicates that an option of this type MUST NOT be present. The resolution of Href.* options against a base URI is specified in Section 5.

	Absolute	Relative	Default	Anonymous
Href.Scheme	yes	no	no	no
Href.Host.Name	yes	no	no	no
Href.Host.IPv4	yes	no	no	no
Href.Host.IPv6	yes	no	no	no
Href.Port	yes	no	no	no
Href.Path	yes	yes	no	no
Href.Query	yes	yes	no	no
Href.Fragment	yes	yes	no	no

Table 2: Permitted Href.* Options by Href Type

4.4. Id

The Id Option specifies an unique identifier for the link or form that can be used as a fragment identifier for this link or form in the CoRAL representation. The option value must be unique amongst

all the IDs in the representation. The value must contain at least one character and must not contain any space characters.

4.5. Method

The Method Option indicates the CoAP method to use for form submission. The option value is one of the CoAP method codes defined in the CoAP Method Codes registry.

4.6. Other-Href

In case the target resource URI cannot be expressed with the Href.* Options, the URI can be specified using the Other-Href Option.

The option value is a string that matches the syntax of the URI-reference production defined in [RFC3986]. The Other-Href Option MUST take precedence over any of the Href.* Options, each of which MUST NOT be included in a link or form containing the Other-Href Option.

4.7. Title

The Title Option, when present, is used to label the target of a link such that it can be used as a human-readable identifier (e.g., a menu entry).

5. Reference Resolution

This section defines the process of resolving the sequence of Href.* options in a link or a form to an absolute URI suitable for inclusion in a CoAP request. The URI reference is resolved against a base URI that is determined as specified in Section 5.1 of RFC 3986. The base URI is assumed to be pre-parsed into a sequence of Href.* options; the result is returned as a sequence of Href.* options as well.

The following pseudocode describes an algorithm for transforming a URI reference R into its target URI T, using the Href Type H, the Link or Form Relation Type S and the base URI B.

```
      if (H == 3) then
         T = [ ]
      elif (H == 2) then
         T = [ (k, v) | (k, v) <- B, k <= Href.Path ] ++
             [ (Href.Path, S) ]
      elif (R starts with Href.Scheme) then
         T = R
      elif (R starts with Href.Host.*) then
         T = [ (k, v) | (k, v) <- B, k == Href.Scheme ] ++
             [ (k, v) | (k, v) <- R, k >  Href.Scheme ]
      elif (R starts with Href.Port) then
         T = [ (k, v) | (k, v) <- B, k <  Href.Port ] ++
             [ (k, v) | (k, v) <- R, k >= Href.Port ]
      elif (H == 1) then
         T = [ (k, v) | (k, v) <- B, k <= Href.Path ] ++
             [ (k, v) | (k, v) <- R, k >= Href.Path ]
      else
         T = [ (k, v) | (k, v) <- B, k <  Href.Path ] ++
             [ (k, v) | (k, v) <- R, k >= Href.Path ]
      endif
```

6. Security Considerations

TODO.

7. IANA Considerations

7.1. CoRAL Option Number Registry

This document establishes the CoRAL Option Number registry for the option numbers used in CoRAL. The registry is located within the CoRE Parameters registry.

7.1.1. Registering New Option Numbers

Option numbers are registered on the advice of a Designated Expert (appointed by the IESG or their delegate), with a Specification Required (using terminology from [RFC5226]).

Registration requests consist of the completed registration template below, typically published in an RFC. However, to allow for the allocation of values prior to publication, the Designated Expert may approve registration once they are satisfied that a specification will be published.

The registration template is:

o Option Number:

- Option Name:

- Reference:

7.1.2. Initial Registry Contents

The CoRAL Option Number registry's initial contents are:

- Option Number: 0
  Option Name: Id
  Reference: [RFCXXXX]

- Option Number: 1
  Option Name: Method
  Reference: [RFCXXXX]

- Option Number: 2
  Option Name: Accept
  Reference: [RFCXXXX]

- Option Number: 3
  Option Name: Format
  Reference: [RFCXXXX]

- Option Number: 4
  Option Name: Href.Scheme
  Reference: [RFCXXXX]

- Option Number: 6
  Option Name: Href.Host.Name
  Reference: [RFCXXXX]

- Option Number: 7
  Option Name: Href.Host.IPv4
  Reference: [RFCXXXX]

- Option Number: 8
  Option Name: Href.Host.IPv6
  Reference: [RFCXXXX]

- Option Number: 11
  Option Name: Href.Port
  Reference: [RFCXXXX]

- Option Number: 12
  Option Name: Href.Path
  Reference: [RFCXXXX]

o Option Number: 13
  Option Name: Href.Query
  Reference: [RFCXXXX]

o Option Number: 14
  Option Name: Href.Fragment
  Reference: [RFCXXXX]

o Option Number: 15
  Option Name: Other-Href
  Reference: [RFCXXXX]

o Option Number: 20
  Option Name: Title
  Reference: [RFCXXXX]

7.2. Media Type

This document registers the media type "application/coral" in the "Media Types" registry.

Type name:
  application

Subtype name:
  coral

Required parameters:
  N/A

Optional parameters:
  N/A

Encoding considerations:
  CoRAL is a binary encoding.

Security considerations:
  See Section 6 of [RFCXXXX].

Interoperability considerations:
  There are no known interoperability issues.

Published specification:
  [RFCXXXX]

Applications that use this media type:
  N/A

Fragment identifier considerations:
   Fragment identifiers used with "application/coral" representations
   refer to the link or form with the indicated unique identifier.
   See Section 4.4 of [RFCXXXX] for details.

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:
   See "Author's Address" section of [RFCXXXX].

Intended usage:
   COMMON

Restrictions on usage:
   N/A

Author:
   See "Author's Address" section of [RFCXXXX].

Change controller:
   IESG

7.3. Structured Syntax Suffix

This document registers the suffix "+coral" in the "Structured Syntax Suffix" registry.

Name:
   Constrained RESTful Application Language (CoRAL)

+suffix:
   +coral

References:
   [RFCXXXX]

Encoding considerations:
   CoRAL is a binary encoding.

Interoperability considerations:
   There are no known interoperability issues.

Fragment identifier considerations:

The syntax and semantics of fragment identifiers specified for +coral are as specified for "application/coral".

Security considerations:
See Section 6 of [RFCXXXX].

Contact:
See "Author's Address" section of [RFCXXXX].

Author/Change controller:
IESG

7.4. CoAP Content-Format

This document registers a content format for the "application/coral" media type in the "CoAP Content-Formats" registry.

- Media Type: application/coral
  Encoding: -
  ID: 70
  Reference: [RFCXXXX]

8. References

8.1. Normative References

[I-D.hartke-core-apps]
Hartke, K., "CoRE Application Descriptions", draft-hartke-core-apps-03 (work in progress), February 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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://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, <http://www.rfc-editor.org/info/rfc7252>.

8.2. Informative References

[FOAF] Brickley, D. and L. Miller, "FOAF Vocabulary Specification 0.99", January 2014, <http://xmlns.com/foaf/spec/20140114.html>.

[I-D.greevenbosch-appsawg-cbor-cddl]
Vigano, C. and H. Birkholz, "CBOR data definition language (CDDL): a notational convention to express CBOR data structures", draft-greevenbosch-appsawg-cbor-cddl-07 (work in progress), October 2015.

[I-D.hartke-core-lighting]
Hartke, K., "CoRE Lighting", draft-hartke-core-lighting-00 (work in progress), September 2015.

[I-D.kelly-json-hal]
Kelly, M., "JSON Hypertext Application Language", draft-kelly-json-hal-07 (work in progress), July 2015.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

[W3C.REC-rdf11-concepts-20140225]
Cyganiak, R., Wood, D., and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax", World Wide Web Consortium Recommendation REC-rdf11-concepts-20140225, February 2014, <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225>.

[W3C.REC-turtle-20140225]
Prud&#039;hommeaux, E. and G. Carothers, "RDF 1.1 Turtle", World Wide Web Consortium Recommendation REC-turtle-20140225, February 2014, <http://www.w3.org/TR/2014/REC-turtle-20140225>.

[W3C.REC-webarch-20041215]
Jacobs, I. and N. Walsh, "Architecture of the World Wide Web, Volume One", World Wide Web Consortium Recommendation REC-webarch-20041215, December 2004, <http://www.w3.org/TR/2004/REC-webarch-20041215>.

[WOTPRAC] Peintner, D. and M. Kovatsch, "WoT Current Practices", February 2016, <http://w3c.github.io/wot/current-practices/wot-practices.html>.

Appendix A. Examples

A.1. CoRE Lighting

CoRE Lighting [I-D.hartke-core-lighting] defines a benchmark scenario for the exploration of hypermedia-oriented design in constrained, RESTful environments. The bulletin board example presented in Section 5.2.1 of [I-D.hartke-core-lighting] could be serialized in CoRAL as follows:

```
   [ /rel_form/           68,
     /create-item/         1,
     [ /accept/            2, 65200,
       /href.path/        12, "bulletins" ]]
   [ /abs_link/            0,
     /item/               30,
     [ /format/            3, 65200,
       /href.host.name/    6, "light-bulb.example" ],
     <<<
       [ /rel_link/        4,
         /config/       -100,
         [ /format/        3, 65202,
           /href.path/    12, "config" ]],
       [ /anon_link/      12,
         /name/         -101,
         [ /format/        3, 0 /text//plain/ ],
         <<<
           Light 2
         >>> ],
       [ /anon_link/      12,
         /purpose/      -102,
         [ /format/        3, 0 /text//plain/ ],
         <<<
           Illuminates the couch.
         >>> ],
       [ /anon_link/      12,
         /location/     -103,
         [ /format/        3, 0 /text//plain/ ],
```

```
            <<<
              Living Room
            >>> ]
        >>> ]
      [ /abs_link/            0,
        /item/               30,
        [ /format/            3, 65200,
          /href.host.name/    6, "remote-control.example" ],
        <<<
          [ /rel_link/        4,
            /about/           1,
            [ /format/        3, 65203,
              /href.path/    12, "state" ]],
          [ /anon_link/      12,
            /name/         -101,
            [ /format/        3, 0 /text//plain/ ],
            <<<
              LRC 1
            >>> ],
          [ /anon_link/      12,
            /purpose/      -102,
            [ /format/        3, 0 /text//plain/ ],
            <<<
              Controls Light 2.
            >>> ],
          [ /anon_link/      12,
            /location/     -103,
            [ /format/        3, 0 /text//plain/ ],
            <<<
              Living Room
            >>> ]
        >>> ]
```

A.2. W3C WoT Thing Description

The W3C Web of Things (WoT) Thing Description (TD) [WOTPRAC] provides a vocabulary for describing an a 'thing' based on metadata and interactions. A thing description like

```
      {
         "@context": ".../w3c-wot-td-context.jsonld",
         "interactions": [
            {
               "@type": "Property",
               "name": "colorTemperature",
               "outputData": "xsd:unsignedShort",
               "writable": true
            }, {
               "@type": "Property",
               "name": "rgbValueRed",
               "outputData": "xsd:unsignedByte",
               "writable": false
            }, {
               "@type": "Property",
               "name": "rgbValueGreen",
               "outputData": "xsd:unsignedByte",
               "writable": false
            }, {
               "@type": "Property",
               "name": "rgbValueBlue",
               "outputData": "xsd:unsignedByte",
               "writable": false
            }
         ]
      }
```

could be serialized in CoRAL as follows:

```
      [ /updateable_def_link/   10,
        /colorTemperature/    -200,
        [ /format/               3, 10 /uint16/ ]]
      [ /def_link/               8,
        /rgbValueRed/         -201,
        [ /format/               3, 9 /uint8/ ]]
      [ /def_link/               8,
        /rgbValueGreen/       -202,
        [ /format/               3, 9 /uint8/ ]]
      [ /def_link/               8,
        /rgbValueBlue/        -203,
        [ /format/               3, 9 /uint8/ ]]
```

Each "Property" interaction in Thing Description is mapped to a link, the "name" attribute to a local link relation type, the "outputData" attribute to the Format Option, and the "writable" attribute to the Updateable Flag.

(This example assumes the definition of appropriate local link relations, a media type with content format ID 9 for xsd:unsignedByte and a media type with content format ID 10 for xsd:unsignedShort.)

Acknowledgements

This specification is heavily inspired by the JSON Hypertext Application Language (HAL) [I-D.kelly-json-hal]; the author of and contributors to that specification are acknowledged for their great work.

Yassin Nasir Hassan suggested placing the hypermedia controls for modifying a link target in the link context rather than in the representation of the link target.

Carsten Bormann contributed the CDDL grammar and CBOR examples.

Author's Address

Klaus Hartke
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63905
Email: hartke@tzi.org