Constrained Resource Identifiers

Abstract

   The Constrained Resource Identifier (CRI) is a complement to the
   Uniform Resource Identifier (URI) that serializes the URI components
   in Concise Binary Object Representation (CBOR) instead of a sequence
   of characters. This simplifies parsing, comparison and reference
   resolution in environments with severe limitations on processing
   power, code size, and memory size.

Discussion Venues

   This note is to be removed before publishing as an RFC.

   Discussion of this document takes place on the Constrained RESTful
   Environments Working Group mailing list (core@ietf.org), which is
   archived at https://mailarchive.ietf.org/arch/browse/core/. Source
   for this draft and an issue tracker can be found at https://
   github.com/core-wg/href

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 28 April 2022.

Copyright Notice

Table of Contents

1. **Introduction**

The Uniform Resource Identifier (URI) [RFC3986] and its most common
usage, the URI reference, are the Internet standard for linking to
resources in hypertext formats such as HTML [W3C.REC-
html52-20171214] or the HTTP "Link" header field [RFC8288].

A URI reference is a sequence of characters chosen from the
repertoire of US-ASCII characters. The individual components of a
URI reference are delimited by a number of reserved characters,
which necessitates the use of a character escape mechanism called
"percent-encoding" when these reserved characters are used in a non-

delimiting function. The resolution of URI references involves parsing a character sequence into its components, combining those components with the components of a base URI, merging path components, removing dot-segments, and recomposing the result back into a character sequence.

Overall, the proper handling of URI references is quite intricate. This can be a problem especially in constrained environments [RFC7228], where nodes often have severe code size and memory size limitations. As a result, many implementations in such environments support only an ad-hoc, informally-specified, bug-ridden, non-interoperable subset of half of RFC 3986.

This document defines the *Constrained Resource Identifier (CRI)* by constraining URIs to a simplified subset and serializing their components in Concise Binary Object Representation (CBOR) [RFC8949] instead of a sequence of characters. This allows typical operations on URI references such as parsing, comparison and reference resolution (including all corner cases) to be implemented in a comparatively small amount of code.

As a result of simplification, however, CRIs are not capable of expressing all URIs permitted by the generic syntax of RFC 3986 (hence the "constrained" in "Constrained Resource Identifier"). The supported subset includes all URIs of the Constrained Application Protocol (CoAP) [RFC7252], most URIs of the Hypertext Transfer Protocol (HTTP) [RFC7230], Uniform Resource Names (URNs) [RFC8141], and other similar URIs. The exact constraints are defined in Section 2.

1.1. Notational Conventions

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.

In this specification, the term "byte" is used in its now customary sense as a synonym for "octet".

Terms defined in this document appear in *cursive* where they are introduced (rendered in plain text as the new term surrounded by underscores).

2. Constraints

A Constrained Resource Identifier consists of the same five components as a URI: scheme, authority, path, query, and fragment. The components are subject to the following constraints:

C1.
The scheme name can be any Unicode string (see Definition D80 in [Unicode]) that matches the syntax of a URI scheme (see Section 3.1 of [RFC3986], which constrains schemes to ASCII) and is lowercase (see Definition D139 in [Unicode]). The scheme is always present.

C2.
An authority is always a host identified by an IP address or registered name, along with optional port information. User information is not supported.

Alternatively, the authority can be absent; the two cases for this defined in Section 3.3 of [RFC3986] are modeled by two different values used in place of an absent authority:

* the path can begin with a root ("/", as when the authority is present), or

* the path can be rootless.

C3.
An IP address can be either an IPv4 address or an IPv6 address, optionally with a zone identifier [RFC6874]. Future versions of IP are not supported.

C4.
A registered name is a sequence of one or more *labels*, which, when joined with dots (".") in between them, result in a Unicode string that is lowercase and in Unicode Normalization Form C (NFC) (see Definition D120 in [Unicode]). (The syntax may be further restricted by the scheme.)

C5.
A port is always an integer in the range from 0 to 65535. Ports outside this range, empty ports (port subcomponents with no digits, see Section 3.2.3 of [RFC3986]), or ports with redundant leading zeros, are not supported.

C6.
The port is omitted if and only if the port would be the same as the scheme's default port (provided the scheme is defining such a default port) or the scheme is not using ports.

C7.
A path consists of zero or more path segments. A path must not consist of a single zero-length path segment, which is considered equivalent to a path of zero path segments.

C8.
A path segment can be any Unicode string that is in NFC, with the exception of the special "." and ".." complete path segments. It can be the zero-length string. No special constraints are placed on the first path segment.

C9.
A query always consists of one or more query parameters. A query parameter can be any Unicode string that is in NFC. It

is often in the form of a "key=value" pair. When converting a CRI to a URI, query parameters are separated by an ampersand ("&") character. (This matches the structure and encoding of the target URI in CoAP requests.) Queries are optional; there is a difference between an absent query and a single query parameter that is the empty string.

C10. A fragment identifier can be any Unicode string that is in NFC. Fragment identifiers are optional; there is a difference between an absent fragment identifier and a fragment identifier that is the empty string.

C11. The syntax of registered names, path segments, query parameters, and fragment identifiers may be further restricted and sub-structured by the scheme. There is no support, however, for escaping sub-delimiters that are not intended to be used in a delimiting function.

C12. When converting a CRI to a URI, any character that is outside the allowed character range or is a delimiter in the URI syntax is percent-encoded. For CRIs, percent-encoding always uses the UTF-8 encoding form (see Definition D92 in [Unicode]) to convert the character to a sequence of bytes (that is then converted to a sequence of %HH triplets).

2.1. Constraints by example

While most URIs in everyday use can be converted to CRIs and back to URIs matching the input after syntax-based normalization of the URI, these URIs illustrate the constraints by example:

  *https://host%ffname, https://example.com/x?data=%ff

  All URI components must, after percent decoding, be valid UTF-8 encoded text. Bytes that are not valid UTF-8 show up, for example, in BitTorrent web seeds.

  *https://example.com/component%3bone;component%3btwo, http://example.com/component%3dequals

  While delimiters can be used in an escaped and unescaped form in URIs with generally distinct meanings, CRIs only support one escapable delimiter character per component, which is the delimiter by which the component is split up in the CRI.

  Note that the separators . (for authority parts), / (for paths), & (for query parameters) are special in that they are syntactic delimiters of their respective components in CRIs. Thus, the following examples *are* convertible to CRIs:

https://interior%2edot/

       https://example.com/path%2fcomponent/second-component

       https://example.com/x?ampersand=%26&questionmark=?

     *https://alice@example.com/

       The user information can not be expressed in CRIs.

     *URIs with an authority but a completely empty path (eg. http://
      example.com)

       CRIs with an authority component always produce at least a slash
       in the path component.

       For generic schemes, the conversion of scheme://example.com to a
       CRI is impossible because no CRI produces a URI with an authority
       not followed by a slash following the rules of [Section 6.1](#). Most
       schemes do not distinguish between the empty path and the path
       containing a single slash when an authority is set (as
       recommended in [RFC3986]). For these schemes, that equivalence
       allows converting even the slash-less URI to a CRI (which, when
       converted back, produces a slash after the authority).

2.2. Constraints not expressed by the data model

   There are syntactically valid CRIs and CRI references that can not
   be converted into a URI or URI reference, respectively.

   For CRI references, this is acceptable -- they can be resolved still
   and result in a valid CRI that can be converted back. (An example of
   this is [0, ["p"]] which appends a slash and the path segment "p" to
   its base).

   (Full) CRIs that do not correspond to a valid URI are not valid on
   their own, and can not be used. Normatively they are characterized
   by the [Section 6.1](#) process producing a valid and syntax-normalized
   URI. For easier understanding, they are listed here:

     *CRIs (and CRI references) containing a path component "." or
      "..".

       These would be removed by the remove_dot_segments algorithm of
       [RFC3986], and thus never produce a normalized URI after
       resolution.

       (In CRI references, the discard value is used to afford segment
       removal, and with "." being an unreserved character, expressing

them as "%2e" and "%2e%2e" is not even viable, let alone
practical).

   *CRIs without authority whose path starts with two or more empty
   segments.

   When converted to URIs, these would violate the requirement that
   in absence of an authority, a URI's path can not begin with two
   slash characters, and they would be indistinguishable from a URI
   with a shorter path and a present but empty authority component.

3. Creation and Normalization

In general, resource identifiers are created on the initial creation
of a resource with a certain resource identifier, or the initial
exposition of a resource under a particular resource identifier.

A Constrained Resource Identifier **SHOULD** be created by the naming
authority that governs the namespace of the resource identifier (see
also [RFC8820]). For example, for the resources of an HTTP origin
server, that server is responsible for creating the CRIs for those
resources.

The naming authority **MUST** ensure that any CRI created satisfies the
constraints defined in Section 2. The creation of a CRI fails if the
CRI cannot be validated to satisfy all of the constraints.

If a naming authority creates a CRI from user input, it **MAY** apply
the following (and only the following) normalizations to get the CRI
more likely to validate:

   *map the scheme name to lowercase (C1);

   *map the registered name to NFC (C4) and split it on embedded
    dots;

   *elide the port if it is the default port for the scheme (C6);

   *elide a single zero-length path segment (C7);

   *map path segments, query parameters and the fragment identifier
    to NFC (C8, C9, C10).

Once a CRI has been created, it can be used and transferred without
further normalization. All operations that operate on a CRI **SHOULD**
rely on the assumption that the CRI is appropriately pre-normalized.
(This does not contradict the requirement that when CRIs are
transferred, recipients must operate on as-good-as untrusted input
and fail gracefully in the face of malicious inputs.)

4. **Comparison**

    One of the most common operations on CRIs is comparison: determining
    whether two CRIs are equivalent, without dereferencing the CRIs
    (using them to access their respective resource(s)).

    Determination of equivalence or difference of CRIs is based on
    simple component-wise comparison. If two CRIs are identical
    component-by-component (using code-point-by-code-point comparison
    for components that are Unicode strings) then it is safe to conclude
    that they are equivalent.

    This comparison mechanism is designed to minimize false negatives
    while strictly avoiding false positives. The constraints defined in
    Section 2 imply the most common forms of syntax- and scheme-based
    normalizations in URIs, but do not comprise protocol-based
    normalizations that require accessing the resources or detailed
    knowledge of the scheme's dereference algorithm. False negatives can
    be caused, for example, by CRIs that are not appropriately pre-
    normalized and by resource aliases.

    When CRIs are compared to select (or avoid) a network action, such
    as retrieval of a representation, fragment components (if any)
    should be excluded from the comparison.

5. **CRI References**

    The most common usage of a Constrained Resource Identifier is to
    embed it in resource representations, e.g., to express a hyperlink
    between the represented resource and the resource identified by the
    CRI.

    This section defines the serialization of CRIs in Concise Binary
    Object Representation (CBOR) [RFC8949]. To reduce representation
    size, CRIs are not serialized directly. Instead, CRIs are indirectly
    referenced through *CRI references*. These take advantage of
    hierarchical locality and provide a very compact encoding. The CBOR
    serialization of CRI references is specified in Section 5.1.

    The only operation defined on a CRI reference is *reference
    resolution*: the act of transforming a CRI reference into a CRI. An
    application **MUST** implement this operation by applying the algorithm
    specified in Section 5.3 (or any algorithm that is functionally
    equivalent to it).

    The reverse operation of transforming a CRI into a CRI reference is
    unspecified; implementations are free to use any algorithm as long
    as reference resolution of the resulting CRI reference yields the
    original CRI. Notably, a CRI reference is not required to satisfy

all of the constraints of a CRI; the only requirement on a CRI
reference is that reference resolution **MUST** yield the original CRI.

When testing for equivalence or difference, applications **SHOULD NOT**
directly compare CRI references; the references should be resolved
to their respective CRI before comparison.

5.1. CBOR Serialization

A CRI reference is encoded as a CBOR array [RFC8949], with the
structure as described in the Concise Data Definition Language
(CDDL) [RFC8610] as follows:

```
; not expressed in this CDDL spec: trailing nulls to be left off

CRI-Reference = [
  ((scheme / null, authority / no-authority)
   // discard),                    ; relative reference
  path / null,
  query / null,
  fragment / null
]

scheme      = scheme-name / scheme-id
scheme-name = text .regexp "[a-z][a-z0-9+.-]*"
scheme-id   = (COAP / COAPS / HTTP / HTTPS / other-scheme)
              .within nint
COAP = -1 COAPS = -2 HTTP = -3 HTTPS = -4
other-scheme = nint .feature "scheme-id-extension"

no-authority = NOAUTH-NOSLASH / NOAUTH-LEADINGSLASH
NOAUTH-NOSLASH = null
NOAUTH-LEADINGSLASH = true

authority   = [host, ?port]
host        = (host-name // host-ip)
host-name   = (*text) ; lowercase, NFC labels
host-ip     = (bytes .size 4 //
                (bytes .size 16, ?zone-id))
zone-id     = text
port        = 0..65535

discard     = DISCARD-ALL / 0..127
DISCARD-ALL = true
path        = [*text]
query       = [*text]
fragment    = text
```

This CDDL specification is simplified for exposition and needs to be augmented by the following rule for interchange: Trailing null values are removed, and two leading null values (scheme and authority both not given) are represented by using the discard alternative instead. A complete CDDL specification is given in Appendix A.

The rules scheme, authority, path, query, fragment correspond to the (sub-)components of a CRI, as described in Section 2, with the addition of the discard section. The discard section can be used when neither a scheme nor an authority is present. It then expresses path prefixes such as "/", "./", "../", "../../", etc. The exact semantics of the section values are defined by Section 5.3.

Most URI references that Section 4.2 of [RFC3986] calls "relative references" (i.e., references that need to undergo a resolution process to obtain a URI) correspond to the CRI form that starts with discard. The exception are relative references with an authority (called a "network-path reference" in Section 4.2 of [RFC3986]), which in CRI references never carry a discard section (the value of discard defaults to true).

The structure of a CRI is visualized using the somewhat limited means of a railroad diagram below.
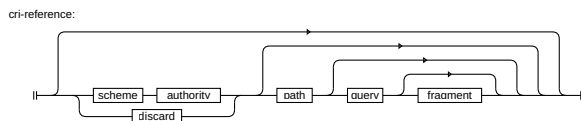
cri-reference:

This visualization does not go into the details of the elements.

Examples:

```
[-1,              / scheme -- equivalent to "coap" /
 [h'C6336401',   / host /
  61616],        / port /
 [".well-known", / path /
  "core"]
]
```

```
[true,                    / discard /
 [".well-known",         / path /
  "core"],
 ["rt=temperature-c"]]   / query /
```

A CRI reference is considered *well-formed* if it matches the CDDL
structure.

A CRI reference is considered *absolute* if it is well-formed and the
sequence of sections starts with a non-null scheme.

A CRI reference is considered *relative* if it is well-formed and the
sequence of sections is empty or starts with a section other than
those that would constitute a scheme.

5.2. Ingesting and encoding a CRI Reference

From an abstract point of view, a CRI Reference is a data structure
with six sections:

scheme, authority, discard, path, query, fragment

Each of these sections can be unset ("null"), except for discard,
which is always an unsigned number or true. If scheme and/or
authority are non-null, discard must be true.

When ingesting a CRI Reference that is in the transfer form, those
sections are filled in from the transfer form (unset sections are
filled with null), and the following steps are performed:

  *If the array is entirely empty, replace it with [0].

  *If discard is present in the transfer form (i.e., the outer array
   starts with true or an unsigned number), set scheme and authority
   to null.

  *If scheme and/or authority are present in the transfer form
   (i.e., the outer array starts with null, a text string, or a
   negative integer), set discard to true.

Upon encoding the abstract form into the transfer form, the inverse
processing is performed: If scheme and/or authority are not null,
the discard value is not transferred (it must be true in this case).
If they are both null, they are both left out and only discard is
transferred. Trailing null values are removed from the array. As a
special case, an empty array is sent in place for a remaining [0]
(URI "").

5.3. Reference Resolution

The term "relative" implies that a "base CRI" exists against which
the relative reference is applied. Aside from fragment-only

references, relative references are only usable when a base CRI is
known.

The following steps define the process of resolving any well-formed
CRI reference against a base CRI so that the result is a CRI in the
form of an absolute CRI reference:

1. Establish the base CRI of the CRI reference and express it in
   the form of an abstract absolute CRI reference.

2. Initialize a buffer with the sections from the base CRI.

3. If the value of discard is true in the CRI reference, replace
   the path in the buffer with the empty array, unset query and
   fragment, and set a true authority to null. If the value of
   discard is an unsigned number, remove as many elements from the
   end of the path array; if it is non-zero, unset query and
   fragment. Set discard to true in the buffer.

4. If the path section is set in the CRI reference, append all
   elements from the path array to the array in the path section
   in the buffer; unset query and fragment.

5. Apart from the path and discard, copy all non-null sections
   from the CRI reference to the buffer in sequence; unset
   fragment if query is non-null and thus copied.

6. Return the sections in the buffer as the resolved CRI.

6. Relationship between CRIs, URIs and IRIs

   CRIs are meant to replace both Uniform Resource Identifiers (URIs)
   [RFC3986] and Internationalized Resource Identifiers (IRIs)
   [RFC3987] in constrained environments [RFC7228]. Applications in
   these environments may never need to use URIs and IRIs directly,
   especially when the resource identifier is used simply for
   identification purposes or when the CRI can be directly converted
   into a CoAP request.

   However, it may be necessary in other environments to determine the
   associated URI or IRI of a CRI, and vice versa. Applications can
   perform these conversions as follows:

   CRI to URI
      A CRI is converted to a URI as specified in Section 6.1.

   URI to CRI
      The method of converting a URI to a CRI is unspecified;
      implementations are free to use any algorithm as long as

converting the resulting CRI back to a URI yields an equivalent
URI.

CRI to IRI
> A CRI can be converted to an IRI by first converting it to a URI
> as specified in Section 6.1, and then converting the URI to an
> IRI as described in Section 3.2 of [RFC3987].

IRI to CRI
> An IRI can be converted to a CRI by first converting it to a URI
> as described in Section 3.1 of [RFC3987], and then converting the
> URI to a CRI as described above.

Everything in this section also applies to CRI references, URI
references and IRI references.

6.1. Converting CRIs to URIs

Applications **MUST** convert a CRI reference to a URI reference by
determining the components of the URI reference according to the
following steps and then recomposing the components to a URI
reference string as specified in Section 5.3 of [RFC3986].

scheme
> If the CRI reference contains a scheme section, the scheme
> component of the URI reference consists of the value of that
> section. Otherwise, the scheme component is unset.

authority
> If the CRI reference contains a host-name or host-ip item, the
> authority component of the URI reference consists of a host
> subcomponent, optionally followed by a colon (":") character and
> a port subcomponent. Otherwise, the authority component is unset.
>
> The host subcomponent consists of the value of the host-name or
> host-ip item.
>
> The host-name is turned into a single string by joining the
> elements separated by dots ("."). Any character in the value of a
> host-name item that is not in the set of unreserved characters
> (Section 2.3 of [RFC3986]) or "sub-delims" (Section 2.2 of
> [RFC3986]) **MUST** be percent-encoded.
>
> The value of a host-ip item **MUST** be represented as a string that
> matches the "IPv4address" or "IP-literal" rule (Section 3.2.2 of
> [RFC3986]). Any zone-id is appended to the string, separated by
> "%25" as defined in Section 2 of [RFC6874], or as specified in a
> successor zone-id specification document; this also leads to a
> modified "IP-literal" rule as specified in these documents.

If the CRI reference contains a port item, the port subcomponent
consists of the value of that item in decimal notation.
Otherwise, the colon (":") character and the port subcomponent
are both omitted.

path

If the CRI reference contains a discard item of value true, the
path component is prefixed by a slash ("/") character. If it
contains a discard item of value 0 and the path item is present,
the conversion fails. Otherwise, the path component is prefixed
by as many "../" components as the discard value minus one
indicates.

If the discard item is not present and the CRI reference contains
an authority that is true, the path component of the URI
reference is prefixed by the zero-length string. Otherwise, the
path component is prefixed by a slash ("/") character.

If the CRI reference contains one or more path items, the prefix
is followed by the value of each item, separated by a slash ("/")
character.

Any character in the value of a path item that is not in the set
of unreserved characters or "sub-delims" or a colon (":") or
commercial at ("@") character **MUST** be percent-encoded.

If the authority component is present (not null or true) and the
path component does not match the "path-abempty" rule
(Section 3.3 of [RFC3986]), the conversion fails.

If the authority component is not present, but the scheme
component is, and the path component does not match the "path-
absolute", "path-rootless" (authority == true) or "path-empty"
rule (Section 3.3 of [RFC3986]), the conversion fails.

If neither the authority component nor the scheme component are
present, and the path component does not match the "path-
absolute", "path-noscheme" or "path-empty" rule (Section 3.3 of
[RFC3986]), the conversion fails.

query

If the CRI reference contains one or more query items, the query
component of the URI reference consists of the value of each
item, separated by an ampersand ("&") character. Otherwise, the
query component is unset.

Any character in the value of a query item that is not in the set
of unreserved characters or "sub-delims" or a colon (":"),
commercial at ("@"), slash ("/") or question mark ("?") character

MUST be percent-encoded. Additionally, any ampersand character
("&") in the item value MUST be percent-encoded.

fragment
   If the CRI reference contains a fragment item, the fragment
   component of the URI reference consists of the value of that
   item. Otherwise, the fragment component is unset.

   Any character in the value of a fragment item that is not in the
   set of unreserved characters or "sub-delims" or a colon (":"),
   commercial at ("@"), slash ("/") or question mark ("?") character
   MUST be percent-encoded.

7. Implementation Status

   With the exception of the authority=true fix and host-names split
   into labels, CRIs are implemented in https://gitlab.com/chrysn/
   micrurus.

8. Security Considerations

   Parsers of CRI references must operate on input that is assumed to
   be untrusted. This means that parsers MUST fail gracefully in the
   face of malicious inputs. Additionally, parsers MUST be prepared to
   deal with resource exhaustion (e.g., resulting from the allocation
   of big data items) or exhaustion of the call stack (stack overflow).
   See Section 10 of [RFC8949] for additional security considerations
   relating to CBOR.

   The security considerations discussed in Section 7 of [RFC3986] and
   Section 8 of [RFC3987] for URIs and IRIs also apply to CRIs.

9. IANA Considerations

   This document has no IANA actions.

10. References

10.1. 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>.

   [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, <https://
              www.rfc-editor.org/info/rfc3986>.

[RFC3987]   Duerst, M. and M. Suignard, "Internationalized Resource
            Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987,
            January 2005, <https://www.rfc-editor.org/info/rfc3987>.

[RFC6874]   Carpenter, B., Cheshire, S., and R. Hinden, "Representing
            IPv6 Zone Identifiers in Address Literals and Uniform
            Resource Identifiers", RFC 6874, DOI 10.17487/RFC6874,
            February 2013, <https://www.rfc-editor.org/info/rfc6874>.

[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>.

[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>.

[RFC8949]   Bormann, C. and P. Hoffman, "Concise Binary Object
            Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/
            RFC8949, December 2020, <https://www.rfc-editor.org/info/
            rfc8949>.

[Unicode]   The Unicode Consortium, "The Unicode Standard, Version
            13.0.0", ISBN 978-1-936213-26-9, March 2020, <https://
            www.unicode.org/versions/Unicode13.0.0/>.

10.2. Informative References

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

[RFC7230]   Fielding, R., Ed. and J. Reschke, Ed., "Hypertext
            Transfer Protocol (HTTP/1.1): Message Syntax and
            Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014,
            <https://www.rfc-editor.org/info/rfc7230>.

[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>.

[RFC8141]   Saint-Andre, P. and J. Klensin, "Uniform Resource Names
            (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017,
            <https://www.rfc-editor.org/info/rfc8141>.

[RFC8288]
        Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/ RFC8288, October 2017, <https://www.rfc-editor.org/info/ rfc8288>.

[RFC8820] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 8820, DOI 10.17487/RFC8820, June 2020, <https://www.rfc-editor.org/info/rfc8820>.

[W3C.REC-html52-20171214] Faulkner, S., Eicholz, A., Leithead, T., Danilo, A., and S. Moon, "HTML 5.2", World Wide Web Consortium Recommendation REC-html52-20171214, 14 December 2017, <https://www.w3.org/TR/2017/REC-html52-20171214>.

Appendix A. CDDL specification

The full CDDL specification is somewhat redundant internally in order to express trailing null suppression.

```
; expressing null suppression

CRI-Reference = [
  ?( ((scheme, (authority / null / true)
      // (null, authority))
    // discard),                    ; relative reference
    ?( (path / null, query / null, fragment) //
      (path / null, query) //
        path)
  )
]

scheme      = scheme-name / scheme-id
scheme-name = text .regexp "[a-z][a-z0-9+.-]*"
scheme-id   = (COAP / COAPS / HTTP / HTTPS / other-scheme)
              .within nint
COAP = -1 COAPS = -2 HTTP = -3 HTTPS = -4
other-scheme = nint .feature "scheme-id-extension"

authority   = [host, ?port]
host        = (host-name // host-ip)
host-name   = (*text) ; lowercase, NFC labels
host-ip     = (bytes .size 4 //
                (bytes .size 16, ?zone-id))
zone-id     = text
port        = 0..65535

discard     = true / 0..127
path        = [*text]
query       = [*text]
fragment    = text
```

Appendix B. Change Log

This section is to be removed before publishing as an RFC.

Changes from -05 to -06

*rework authority:

-split reg-names at dots;

-add optional zone identifiers [RFC6874] to IP addresses

Changes from -04 to -05

*Simplify CBOR structure.

*Add implementation status section.

  Changes from -03 to -04:

     *Minor editorial improvements.

     *Renamed path.type/path-type to discard.

     *Renamed option to section, substructured into items.

     *Simplied the table "resolution-variables".

     *Use the CBOR structure inspired by Jim Schaad's proposals.

  Changes from -02 to -03:

     *Expanded the set of supported schemes (#3).

     *Specified creation, normalization and comparison (#9).

     *Clarified the default value of the path.type option (#33).

     *Removed the append-relation path.type option (#41).

     *Renumbered the remaining path.types.

     *Renumbered the option numbers.

     *Restructured the document.

     *Minor editorial improvements.

  Changes from -01 to -02:

     *Changed the syntax of schemes to exclude upper case characters
      (#13).

     *Minor editorial improvements (#34 #37).

  Changes from -00 to -01:

     *None.

Acknowledgements

Thanks to Christian Amsüss, Ari Keränen, Jim Schaad and Dave Thaler for helpful comments and discussions that have shaped the document.

Contributors

Klaus Hartke
Ericsson
Torshamnsgatan 23
SE-16483 Stockholm
Sweden

Email: klaus.hartke@ericsson.com

Authors' Addresses

Carsten Bormann (editor)
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

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

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany

Email: henk.birkholz@sit.fraunhofer.de