Workgroup: CoRE Working Group
Internet-Draft: draft-ietf-core-href-10
Published: 7 March 2022
Intended Status: Standards Track
Expires: 8 September 2022
Authors: C. Bormann, Ed.
Universität Bremen TZI
H. Birkholz
Fraunhofer SIT

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.

The present revision -10 of this draft contains an experimental addition that allows representing user information (https://alice@chains.example) in the URI authority component. This feature lacks test vectors and implementation experience at the time of writing and requires discussion.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-core-href/.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (mailto: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 8 September 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, and optionally preceded by user information.

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.

(Note that in Figure 1, no-authority is marked as a feature, as not all CRI implementations will support authority-less URIs.)

C3. A userinfo is a text string built out of unreserved characters (Section 2.3 of [RFC3986]) or "sub-delims" (Section 2.2 of [RFC3986]); any other character needs to be percent-encoded (Section 7). Note that this excludes the ":" character, which is commonly deprecated as a way to delimit a cleartext password in a userinfo.

C4. 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 (it is likely that a binary

mapping would be strongly desirable, and that cannot be designed ahead of time, so these versions need to be added as a future extension if needed).

C5. 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. As per Section 3.2.2 of [RFC3986], a registered name can be empty, for which case a scheme can define a default for the host.)

C6. 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.

C7. 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.

C8. A path consists of zero or more path segments. Note that a path of just a single zero-length path segment is allowed -- this is considered equivalent to a path of zero path segments by HTTP and CoAP, but this equivalence does not hold for CRIs in general as they only perform normalization on the Syntax-Based Normalization level (Section 6.2.2 of [RFC3986], not on the scheme-specific Scheme-Based Normalization level (Section 6.2.3 of [RFC3986]).

(A CRI implementation may want to offer scheme-cognizant interfaces, performing this scheme-specific normalization for schemes it knows. The interface could assert which schemes the implementation knows and provide pre-normalized CRIs. This can also relieve the application from removing a lone zero-length path segment before putting path segments into CoAP Options, i.e., from performing the check and jump in item 8 of Section 6.4 of [RFC7252]. See also SP1 in Appendix A.)

C9. A path segment can be any Unicode string that is in NFC, with the exception of the special "." and ".." complete path segments. Note that this includes the zero-length string.

If no authority is present in a CRI, the leading path segment cannot be empty. (See also SP1 in Appendix A.)

C10. 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.

C11. 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.

C12. 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.

C13. 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).

Examples for URIs at or beyond the boundaries of these constraints are in SP2 in Appendix A.

2.1. Constraints not expressed by the data model

There are syntactically valid CRIs and CRI references that cannot 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 cannot 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 cannot 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 (C5) and split it on embedded dots;

* elide the port if it is the default port for the scheme (C7);

* map path segments, query parameters and the fragment identifier to NFC form (C9, C10, C11).

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 or CRI reference is encoded as a CBOR array [RFC8949], with the structure as described in the Concise Data Definition Language (CDDL) [RFC8610] as follows: RFC Ed.: throughout this section, please replace RFC-XXXX with the RFC number of this specification and remove this note.

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

RFC-XXXX-Definitions = [CRI, CRI-Reference]

CRI = [
  scheme,
  authority / no-authority,
  local-part
]


CRI-Reference = [
  ((scheme / null, authority / no-authority)
   // discard),                 ; relative reference
  local-part
]

local-part = (
  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 / URN / DID /
               other-scheme)
              .within nint
COAP = -1 COAPS = -2 HTTP = -3 HTTPS = -4 URN = -5 DID = -6
other-scheme = nint .feature "scheme-id-extension"

no-authority = NOAUTH-NOSLASH / NOAUTH-LEADINGSLASH
NOAUTH-LEADINGSLASH = null .feature "no-authority"
NOAUTH-NOSLASH = true .feature "no-authority"

authority   = [?userinfo, host, ?port]
userinfo    = (false, text .feature "userinfo")
host        = (host-ip // host-name)
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
```

Figure 1: CDDL for CRI CBOR serialization

This CDDL specification is simplified for exposition and needs to be augmented by the following rule for interchange of CRIs and CRI references: Trailing null values **MUST** be removed, and two leading null values (scheme and authority both not given) are represented by using the discard alternative instead.

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.

5.1.1. The discard Section

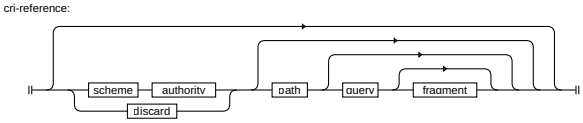
The discard section can be used in a CRI reference when neither a scheme nor an authority is present. It then expresses the operations performed on a base CRI by CRI references that are equivalent to URI references with relative paths and path prefixes such as "/", "./", "../", "../../", etc. "." and ".." are not available in CRIs and are therefore expressed using discard after a normalization step, as is the presence or absence of a leading "/".

E.g., a simple URI reference "foo" specifies to remove one leading segment from the base URI's path, which is represented in the equivalent CRI reference discard section as the value 1; similarly "../foo" removes two leading segments, represented as 2; and "/foo" removes all segments, represented in the discard section as the value true. 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 discard the entire path of the base CRI. These CRI references never carry a discard section: the value of discard defaults to true.

5.1.2. Visualization

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



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

5.1.3. 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 /
```

```
[-6,             / scheme -- equivalent to "did" /
 true,           / authority = NOAUTH-NOSLASH /
 ["web:alice:bob"] / path /
]
```

5.1.4. Specific Terminology

A CRI reference is considered *well-formed* if it matches the structure as expressed in Figure 1 in CDDL, with the additional requirement that trailing null values are removed from the array.

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 (which is implicitly the case when scheme and/or authority are present in the 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 in the buffer if query is non-null in the CRI reference (and therefore has been copied to the buffer).

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, optionally preceded by a userinfo subcomponent. Otherwise, the authority component is unset.

The host subcomponent consists of the value of the host-name or host-ip item.

The userinfo subcomponent, if present, is turned into a single string by appending a "@". Otherwise, both the subcomponent and the "@" sign are omitted. Any character in the value of the userinfo elements 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 host-name is turned into a single string by joining the elements separated by dots ("."). Any character in the elements of a host-name item that is a dot ("."), or 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 superseding zone-id specification document [I-D.carpenter-6man-rfc6874bis]; 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 considered *rooted*. If it contains a discard item of value 0 and the path item is present, the conversion fails. If it contains a positive discard item, the path component is considered *unrooted* and 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 considered unrooted. Otherwise, the path component is considered rooted.

If the CRI reference contains one or more path items, the path component is constructed by concatenating the sequence of representations of these items. These representations generally contain a leading slash ("/") character and the value of each item, processed as discussed below. The leading slash character is omitted for the first path item only if the path component is considered "unrooted".

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. Extended CRI: Accommodating Percent Encoding (PET)

CRIs have been designed to relieve implementations operating on CRIs from string scanning, which both helps constrained implementations and implementations that need to achieve high throughput.

Basic CRI does not support URI components that *require* percent-encoding (Section 2.1 of [RFC3986]) to represent them in the URI syntax, except where that percent-encoding is used to escape the main delimiter in use.

E.g., the URI

```
https://alice/3%2f4-inch
```

is represented by the basic CRI

```
[-4, ["alice"], ["3/4-inch"]]
```

However, percent-encoding that is used at the application level is not supported by basic CRIs:

```
did:web:alice:7%3A1-balun
```

This section presents a method to represent percent-encoded segments of userinfo, hostnames, paths, and queries, as well as fragments.

The four CDDL rules

```
userinfo    = (false, text .feature "userinfo")
host-name   = (*text)
path        = [*text]
query       = [*text]
fragment    = text
```

are replaced with

```
userinfo    = (false, text-or-pet .feature "userinfo")
host-name   = (*text-or-pet)
path        = [*text-or-pet]
query       = [*text-or-pet]
fragment    = text-or-pet

text-or-pet = text /
    text-pet-sequence .feature "extended-cri"

; text1 and pet1 alternating, at least one pet1:
text-pet-sequence = [?text1, ((+(pet1, text1), ?pet1) // pet1)]
; pet is percent-encoded bytes
pet1 = bytes .ne ''
text1 = text .ne ""
```

That is, for each of the host-name, path, and query segments, and for the userinfo and fragment components, an alternate representation is provided besides a simple text string: a non-empty array of alternating non-blank text and byte strings, the text strings of which stand for non-percent-encoded text, while the byte strings retain the special semantics of percent-encoded text without actually being percent-encoded.

The above DID URI can now be represented as:

```
[-6, true, [["web:alice:7", ':', "1-balun"]]]
```

8. Implementation Status

With the exception of the authority=true fix, host-names split into labels, and Section 7, CRIs are implemented in https://gitlab.com/chrysn/micrurus. A golang implementation of version -10 of this document is found at: https://github.com/thomas-fossati/href

9. 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.

10. IANA Considerations

This document has no IANA actions.

11. References

11.1. Normative References

[I-D.carpenter-6man-rfc6874bis] Carpenter, B., Cheshire, S., and R. M. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", Work in Progress, Internet-Draft, draft-carpenter-6man-rfc6874bis-03, 8 February 2022, <https://www.ietf.org/archive/id/draft-carpenter-6man-rfc6874bis-03.txt>.

[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/>.

11.2. 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. The Small Print

This appendix lists a few corner cases of URI semantics that implementers of CRIs need to be aware of, but that are not representative of the normal operation of CRIs.

SP1. Initial (Lone/Leading) Empty Path Segments:

* *Lone empty path segments:* As per [RFC3986], s://x is distinct from s://x/ -- i.e., a URI with an empty path is different from one with a lone empty path segment. However, in HTTP, CoAP, they are implicitly aliased (for CoAP, in item 8 of Section 6.4 of [RFC7252]). As per item 7 of Section 6.5 of [RFC7252], recomposition of a URI without Uri-Path Options from the other URI-related CoAP Options produces s://x/, not s://x -- CoAP prefers the lone empty path segment form. TBD: add similar text for HTTP, if that can be made. After discussing HTTP semantics, Section 6.2.3 of [RFC3986] even states:

> In general, a URI that uses the generic syntax for authority with an empty path should be normalized to a path of "/".

* *Leading empty path segments without authority*: Somewhat related, note also that URIs and URI references that do not carry an authority cannot represent initial empty path segments (i.e., that are followed by further path segments): s://x//foo works, but in a s://foo URI or an (absolute-path) URI reference of the form //foo the double slash would be mis-parsed as leading in to an authority.

SP2. Constraints (Section 2) of CRIs/basic CRIs

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, basic CRIs (i.e., without percent-encoded text Section 7) 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 basic 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 be expressed in CRIs if the "userinfo" feature is present. The URI https://@example.com is represented as [-4, [false, "", "example", "com"]]; the false serves as a marker that the next element is the userinfo.

The rules do not cater for unencoded ":" in userinfo, which is commonly considered a deprecated inclusion of a literal password.

Appendix B. Change Log

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

Changes from -08 to -09

* Identify more esoteric features with a CDDL ".feature".

* Clarify that well-formedness requires removing trailing nulls.

* Fragments can contain PET.

* Percent-encoded text in PET is treated as byte strings.

* URIs with an authority but a completely empty path (e.g., http://example.com): CRIs with an authority component no longer always produce at least a slash in the path component.

  For generic schemes, the conversion of scheme://example.com to a CRI is now possible because CRI produces a URI with an authority not followed by a slash following the updated rules of Section 6.1. Schemes like http and coap 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 implementations to convert the just-a-slash URI to a CRI with a zero length path array (which, however, when converted back, does not produce a slash after the authority).

(Add an appendix "the small print" for more detailed discussion of pesky corner cases like this.)

Changes from -07 to -08

*Fix the encoding of NOAUTH-NOSLASH / NOAUTH-LEADINGSLASH

*Add URN and DID schemes, add example.

*Add PET

*Remove hopeless attempt to encode "remote trailing nulls" rule in CDDL (which is not a transformation language).

Changes from -06 to -07

*More explicitly discuss constraints (Section 2), add examples (Appendix A, Paragraph 6, Item 1).

*Make CDDL more explicit about special simple values.

*Lots of gratuitous changes from XML2RFC redefinition of <tt> semantics.

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.

*Simplified 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

CRIs were developed by Klaus Hartke for use in the Constrained RESTful Application Language (CoRAL). The current author team is completing this work with a view to achieve good integration with the potential use cases, both inside and outside of CoRAL.

Thanks to Christian Amsüss, Thomas Fossati, Ari Keränen, Jim Schaad, Dave Thaler and Marco Tiloca 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