Workgroup: CoRE Working Group
Internet-Draft: draft-ietf-core-href-04
Published: 7 May 2021
Intended Status: Standards Track
Expires: 8 November 2021
Authors: K. Hartke C. Bormann, Ed.
Ericsson Universitaet Bremen TZI

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.

Note to Readers

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

The issues list for this Internet-Draft can be found at <https://github.com/core-wg/coral/labels/href>.

A reference implementation and a set of test vectors can be found at <https://github.com/core-wg/coral/tree/master/binary/python>.

Status of This Memo

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]) and is lowercase (see Definition D139 in [Unicode]).

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.

C3. An IP address can be either an IPv4 address or an IPv6 address. IPv6 scoped addressing zone identifiers and future versions of IP are not supported.

C4. A registered name can be any 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. Empty ports or ports outside this range 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).

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);

* 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 5.2 (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:

```
CRI-Reference = [
  (authority // discard),
  *path,
  ? (([], fragment)              ; include array only if
     //([+query], ?fragment))    ; at least one query and/or fragment
]

authority   = (?scheme, ?(host, ?port))
scheme      = (scheme-name
               // COAP // COAPS // HTTP // HTTPS)
scheme-name = (false, text .regexp "[a-z][a-z0-9+.-]*")
COAP = -1 COAPS = -2 HTTP = -3 HTTPS = -4
host        = (host-name // host-ip)
host-name   = (true, text)
host-ip     = bytes .size 4 / bytes .size 16
port        = 0..65535
discard     = 0..127
path        = text
query       = text
fragment    = text
```

The rules scheme, host, port, path, query, fragment correspond to the (sub-)components of a CRI, as described in Section 2, with the addition of the discard element. While scheme and host can comprise two array elements, we will treat such a combination as a single "element" in the following exposition. (The combination is needed to disambiguate what would otherwise be a leading text string as a scheme, host, or path element.) The discard element or its absence can be used to express path prefixes such as "/", "./", "../", "../../", etc. The exact semantics of the element values are defined by Section 5.2.

Examples:

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

```
[".well-known", / path /
 "core",        / path /
 ["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 elements starts with a scheme.

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

5.2. 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 absolute CRI reference. (The base CRI can be established in a number of ways; see Section 5.1 of [RFC3986].) Assign each element an element number according to the number E for that element in Table 1.

2. Determine the values of two variables, T and E, based on the first element in the sequence of elements of the CRI reference to be resolved, according to Table 1.

3. Initialize a buffer with all the elements from the base CRI where the element number is less than the value of E.

4. If the value of T is greater than 0, remove the last T-many path elements from the end of the buffer (up to the number of path elements in the buffer).

5. Append all the elements from the CRI reference to the buffer, except for any discard element.

6. If the number of path elements in the buffer is one and the value of that element is the zero-length string, remove that element from the buffer.

7. Return the sequence of elements in the buffer as the resolved CRI.

First Element	T	E
(scheme)	0	0
(host)	0	1
(discard)	element value	3
(path)	0	2
(query)	0	3
(fragment)	0	4
none/empty sequence	0	4

Table 1: Values of the Variables T and E

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 element, the scheme component of the URI reference consists of the value of that element. Otherwise, the scheme component is unset.

authority

If the CRI reference contains a host-name or host-ip element, 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 element.

Any character in the value of a host-name element 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 element **MUST** be represented as a string that matches the "IPv4address" or "IP-literal" rule (Section 3.2.2 of [RFC3986]).

If the CRI reference contains a port element, the port subcomponent consists of the value of that element in decimal

notation. Otherwise, the colon (":") character and the port subcomponent are both omitted.

path

If the CRI reference is an empty sequence of elements or starts with a port element, a path element, or a discard element where the value is not 0, the conversion fails.

If the CRI reference contains a host-name element, a host-ip element or a discard element, the path component of the URI reference is prefixed by a slash ("/") character. Otherwise, the path component is prefixed by the zero-length string.

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

Any character in the value of a path element 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 defined and the path component does not match the "path-abempty" rule (Section 3.3 of [RFC3986]), the conversion fails.

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

If the authority component is unset and the scheme component is unset 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 elements, the query component of the URI reference consists of the value of each element, separated by an ampersand ("&") character. Otherwise, the query component is unset.

Any character in the value of a query element 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 element value **MUST** be percent-encoded.

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

Any character in the value of a fragment element 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. 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.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. 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>.

[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/>.

9.2. 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. Change Log

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

Changes from -03 to -04:

* Minor editorial improvements.

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

* Renamed option to element.

* Simplied Table 1.

* 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 discard option (#33).

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

* Renumbered the remaining discards.

* 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.

Authors' Addresses

Klaus Hartke
Ericsson
Torshamnsgatan 23
SE-16483 Stockholm
Sweden

Email: klaus.hartke@ericsson.com

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

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