CoRE Working Group
Internet-Draft
Updates: 7595 (if approved)
Intended status: Standards Track
Expires: 9 January 2026

C. Bormann, Ed.
Universitt Bremen TZI
H. Birkholz
Fraunhofer SIT

8 July 2025

Constrained Resource Identifiers
draft-ietf-core-href-23

Abstract

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

This RFC updates RFC 7595 to add a note on how the "URI Schemes" registry of RFC 7595 cooperates with the "CRI Scheme Numbers" registry created by the present RFC.

// (This "cref" paragraph will be removed by the RFC editor:) The
// present revision 23 attempts to address the AD review comments;
// it is submitted right before the I-D deadline in order to serve as
// discussion input to IETF 123 even though not all discussions have
// completed. In particular, the updated handling of zone
// identifiers requires some additional scrutiny.

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/. Subscribe at https://www.ietf.org/mailman/listinfo/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 9 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The Uniform Resource Identifier (URI) [STD66] 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 (Section 4.1 of RFC 3986 [STD66]). 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 (Section 5 of RFC 3986 [STD66]) 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 "..", see Section 3.3 of RFC 3986 [STD66]), 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][I-D.ietf-iotops-7228bis], 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.

This document defines the _Constrained Resource Identifier (CRI)_ by constraining URIs to a simplified subset and representing their components in Concise Binary Object Representation (CBOR) [STD94] instead of a sequence of characters. Analogously, _CRI references_ are to CRIs what URI references are to URIs.

CRIs and CRI references allow typical operations on URIs and URI references such as parsing, comparison, and reference resolution (including all corner cases) to be implemented in a comparatively small amount of code and to be less prone to bugs and interoperability issues.

As a result of simplification, however, _Simple CRIs_ (i.e., not using CRI extensions, see Section 7) are not capable of expressing all URIs permitted by the generic syntax of [STD66] (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) [STD97], Uniform Resource Names (URNs) [RFC8141], and other similar URIs. The exact constraints are defined in Section 2. CRI extensions (Section 7) can be defined to address some of the constraints and/or to provide more convenient representations for certain areas of application.

This RFC creates a "CRI Scheme Numbers" registry and updates RFC7595 to add a note on how this new registry cooperates with the "URI Schemes" registry that RFC7595 describes.

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 [BCP14] (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 _italics_ where they are introduced (in the plaintext form of this document, they are rendered as the new term surrounded by underscores).

The general structure of data items is shown in the Concise Data Definition Language (CDDL) [RFC8610] including its control extensions [RFC9165]. Specific examples are notated in CBOR Extended Diagnostic Notation (EDN), as originally introduced in Section 8 of RFC 8949 [STD94] and extended in Appendix G of [RFC8610]. ([I-D.ietf-cbor-edn-literals] more rigorously defines and further extends EDN.)

2. From URIs to CRIs: Considerations and Constraints

2.1. The CRI interchange data model

A Constrained Resource Identifier consists of the same five components as a URI: scheme, authority, path, query, and fragment.

Many components of a URI can be "absent", i.e., they are optional. This is not mirrored in CRIs, where all components are part of all CRIs. Some CRI components can have values that are null or empty arrays. By defining a default value for each of certain components, they often can be elided at the tail of the serialized form during interchange. (Note that some subcomponents such as port numbers or userinfo are optional in a CRI as well and therefore can be absent from a CRI.)

In a CRI reference, components can additionally be _not set_ (indicated by interchanging a discard value instead of scheme and authority, or by null for the scheme, path and query components that can otherwise not have that value). (For example, for a CRI reference where authority is either not set or has either of the NOAUTHORITY values, the equivalent URI reference's authority is absent.)

The components are subject to the considerations and constraints listed in this section. Note that CRI extensions can relax constraints; for example, see Section 7.2 for partially relaxing constraint C0.

C0. Text strings in CRIs ("CRI text strings") are CBOR text strings (i.e., in UTF-8 form [STD63]) that represent Unicode strings (see Definition D80 in [Unicode]) in Unicode Normalization Form C (NFC) (see Definition D120 in [Unicode] and specifically Section 3, Paragraph 7).

C1. The scheme name can be any CRI text string that matches the syntax of a URI scheme (see Section 3.1 of RFC 3986 [STD66], which constrains scheme names to a subset of ASCII), in its canonical form. (The canonical form as per Section 3.1 of RFC 3986 [STD66] requires alphabetic characters to be in lowercase.) The scheme is always present.

C2. An authority is always a host identified by an IP address or a "registered name" (see C5 below), along with optional port information, and optionally preceded by user information.

Alternatively, URIs can be formed without an authority. The two cases for this defined in Section 3.3 of RFC 3986 [STD66] are modeled by two different special values used in the CRI authority component:

* the path can be root-based (zero or more path segments that are each started in the URI with "/", as when the authority is present), or

* the path can be rootless, which requires at least one path segment, the first one of which has non-zero length and is not started in the URI with "/".

(Note that, in Figure 2, 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 RFC 3986 [STD66]) or "sub-delims" (Section 2.2 of RFC 3986 [STD66]); any other character needs to be percent-encoded (Section 7.2). 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; see Section 6.1, Paragraph 4). 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 represented as a sequence of one or more lowercase CRI text string _labels_ that do not contain dots (".") (These labels joined with dots (".") in between them result in the CRI equivalent of a URI registered name as per Section 3.2.2 of RFC 3986 [STD66]. The syntax may be further restricted by the scheme. A URI 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 RFC 3986 [STD66]), or ports with redundant leading zeros, are not supported.

C7. If the scheme’s port handling is known to the CRI creator, it is RECOMMENDED to omit the port if and only if the port would be the same as the scheme’s default port (provided the scheme defines 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 RFC 3986 [STD66]), not on the scheme-specific Scheme-Based Normalization level (Section 6.2.3 of RFC 3986 [STD66]).

(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 CRI text string, 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. Queries are optional in URIs; there is a difference between an absent query and a present query that is the empty string. A CRI represents its query component as an array of zero or more query parameters, which are CRI text strings. Zero query parameters (an empty array) is equivalent to a URI where the query is absent; a single query parameter that is the empty string is equivalent to a URI with a present, but empty, query string. A query in a URI is represented in a CRI by splitting its text up on any ampersand ("&") characters into one or more query parameters, which may contain certain characters (including ampersands) that were percent-encoded in the URI. Query parameters are often in the form of a "key=value" pair. When converting a CRI to a URI, one or more query parameters are constructed into a URI query by joining them together with ampersand characters, where certain characters (including ampersands) present in the query parameters are percent-encoded. (This matches the structure and encoding of the target URI in CoAP requests.)

C11. A fragment identifier can be any CRI text string. Fragment identifiers are optional in URIs; in CRIs there is a difference between a null 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, which are 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.

2.2. CRI References: The discard Component

As with URI references and URIs, CRI references are a shorthand for a CRI that is expressed relative to a base CRI. URI and CRI references often _discard_ part or all of the trailing path segments of the base URI or CRI.

In a URI reference, this is expressed by syntax for its path component such as leading special path segments . and .. or a leading slash before giving the path segments to be added at the end of the (now truncated) base URI. For use in CRI references, we instead add in a discard component as an alternative to the scheme and authority components, making the specification of discarding base URI path segments separate from adding new path segments from the CRI reference.

The discarding intent of a CRI reference is thus fully condensed to a single value in its discard component:

* An unsigned integer as the discard component specifies the number of path segments to be discarded from the base CRI (note that this includes the value 0 which cannot be expressed in URI references that then add any path component);

* the value true as the discard component specifies discarding all path segments from the base CRI.

If a scheme or authority is present in a CRI reference, the discard component is implicitly equivalent to a value of true and thus not included in the interchanged data item.

2.3. 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.

CRI references of this kind can be acceptable -- they still can be resolved and result in a valid full CRI that can be converted back. Examples of this are:

* [0, ["p"]]: appends a slash and the path segment "p" to its base, sets the query to an empty array and the fragment to null

* [0, null, []]: leaves the path alone but sets the query to an empty array and the fragment to null

(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 not producing a valid and syntax-normalized URI. For easier understanding, they are listed here:

* CRIs (and CRI references) containing dot-segments (path segment "." or "..").

  These segments would be removed by the remove_dot_segments algorithm of [STD66], and thus never produce a normalized URI after resolution.

  (In CRI references, the discard value is used to afford segment removal (see Section 2.2), 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 a leading empty segment followed by at least one more segment.

  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. (E.g., two leading empty segments would be indistinguishable from a URI with a shorter path and a present but empty authority component.) (Compare C9.)

* CRIs without authority that are rootless and have an empty path component (e.g., ["a", true, []]), which would be indistinguishable from its root-based equivalent (["a", null, []]) as both would have the URI a:.

3. Creation and Normalization

In general, resource identifiers are generated when a resource is initially created or exposed under a certain resource identifier.

The naming authority that creates a Constrained Resource Identifier SHOULD be the authority that governs the namespace of the resource identifier (see also [BCP190]). For example, for the resources of an HTTP origin server, that server is responsible for creating the CRIs for those resources. If the naming authority creates a URI instead that can be obtained as a conversion result from a CRI (Section 6.1) that CRI can be considered to have been created by the naming authority.

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

If a naming authority creates a CRI from user input, it may need to 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 (C0) and split it on embedded dots (C5);

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

* map path segments (C9), query parameters (C10), and the fragment identifier (C11) to NFC form (C0).

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.)

Note that the processing of CRIs does not imply that all the constraints continuously need to be checked and enforced. Specifically, the text normalization constraints (NFC) can be expanded as: The recipient of a CRI MAY reasonably expect the text strings to be in NFC form, but as with any input MUST NOT fail (beyond possibly not being able to process the specific CRI) if they are not. So the onus of fulfilling the expectation is on the original creator of the CRI, not on each processor (including consumer). This consideration extends to the sources the CRI creator uses in building the text strings, which the CRI creator MAY in turn expect to be in NFC form if that expectation is reasonable. See Appendix C of [MNU] for some background.

CRIs have been designed with the objective that, after the above normalization, conversion of two distinct CRIs to URIs do not yield the "same" URI, including equivalence under syntax-based normalization (Section 6.2.2 of RFC 3986 [STD66]), but not including protocol-based normalization. Note that this objective exclusively applies to (full) CRIs, not to CRI references: these need to be resolved relative to a base URI, with results that may be equivalent or not depending on the base.

4. Comparison

One of the most common operations on CRIs is comparison: determining whether two CRIs are equivalent, without dereferencing the CRIs (i.e., 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) do not play a role and typically are 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.

Section 5.1 first defines the representation of CRIs in Concise Binary Object Representation (CBOR) [STD94]. When reduced representation size is desired, CRIs are often not represented directly. Instead, CRIs are indirectly referenced through _CRI references_. These take advantage of hierarchical locality and provide a very compact encoding. The CBOR representation of CRI references also 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 not specified in detail in this document; 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, it is rarely appropriate for applications to directly compare CRI references; instead, the references are typically resolved to their respective CRIs before comparison.

5.1. CBOR Representation

// RFC Ed.: throughout this section, please replace RFC-XXXX with the
// RFC number of this specification and remove this note.

A CRI or CRI reference is encoded as a CBOR array (Major type 4 in Section 3.1 of RFC 8949 [STD94]). Figure 1 has a coarse visualization of the structure of this array, without going into the details of the elements.

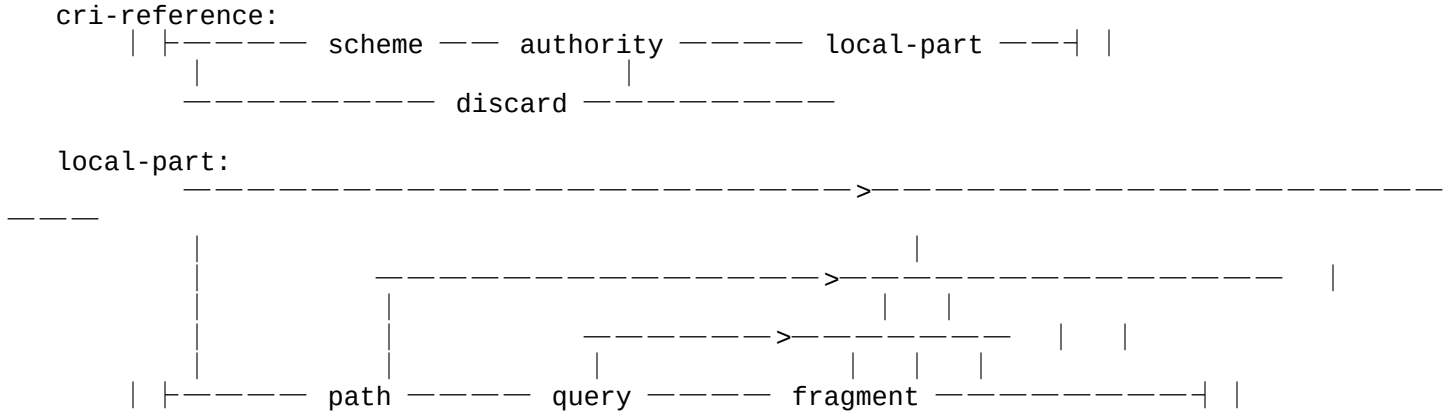

Figure 1: Overall Structure of a CRI or CRI Reference

Figure 2 has a more detailed description of the structure, in CDDL.

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

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

CRI = [
  scheme,
  authority / no-authority,
  path,                    ; use [] for empty path
  query,                   ; use [] for empty query
  fragment / null
]


CRI-Reference = [
  ((scheme / null, authority / no-authority)
   // discard),            ; relative reference
  path / null,             ; null is explicitly not set
  query / null,            ; null is explicitly not set
  fragment / null
]

scheme      = scheme-id / (scheme-name .feature "scheme-name")
scheme-id   = nint               ; -1 - scheme-number
scheme-name = text .regexp "[a-z][a-z0-9+.-]*"

no-authority = NOAUTH-ROOTBASED / NOAUTH-ROOTLESS
NOAUTH-ROOTBASED = null .feature "no-authority"
NOAUTH-ROOTLESS = true .feature "no-authority"

authority   = [?userinfo, host, ?port]
userinfo    = (false, text .feature "userinfo")
host        = (host-ip // host-name)
host-name   = (*text)      ; lowercase, NFC labels; no dot
host-ip     = (bytes .size (4/16), ?zone-id)
zone-id     = text
port        = 0..65535

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

Figure 2: CDDL for CRI CBOR representation

We call the elements of the top-level array _sections_. The sections containing the rules scheme, authority, path, query, fragment correspond to the components of a URI and thus of a CRI, as described

in Section 2. For use in CRI references, the discard section (see also Section 2.2) provides an alternative to the scheme and authority sections.

This CDDL specification is simplified for exposition and needs to be augmented by the following rules for interchange of CRIs and CRI references:

* Trailing default values (Table 1) MUST be removed.

* Two leading null values (scheme and authority both not given) MUST be represented by using the discard alternative instead.

* An empty path in a CRI is represented as the empty array []. Note that for CRI-Reference there is a difference between empty paths and paths that are not set, represented by [] and null, respectively.

* An empty query in a CRI (no query parameters, not even an empty string) is represented as the empty array []; note that this is equivalent to the absence of the question mark in a URI, while the equivalent of just a question mark in a URI is an array with a single query parameter represented by an empty string [""]). Note that for CRI-Reference there is a difference between providing a query as above and a query that is not set, represented by null.

* An empty outer array ([]) is not a valid CRI. It is a valid CRI reference, equivalent to [0] as per Section 5.2, which essentially copies the base CRI up to and including the path section, setting query and fragment to absent.

Section	Default Value
scheme	
authority	null
discard	0
path	[]
query	[]
fragment	null

Table 1: Default Values for CRI Sections

For interchange as separate encoded data items, CRIs MUST NOT use indefinite length encoding (see Section 3.2 of RFC 8949 [STD94]). This requirement is relaxed for specifications that embed CRIs into

an encompassing CBOR representation that does provide for indefinite length encoding; those specifications that are selective in where they provide for indefinite length encoding are RECOMMENDED to not provide it for embedded CRIs.

5.1.1. scheme-name and scheme-id

In the scheme section, a CRI scheme is usually given as a negative integer scheme-id derived from the _scheme number_. Optionally, it can instead be identified by its scheme-name (a text string giving the scheme name as in URIs' scheme section, mapped to lower case). (Note that, in Figure 2, scheme-name is marked as a feature, as only less constrained CRI implementations might support scheme-name.)

Scheme numbers are unsigned integers that are mapped to and from URI scheme names by the "CRI Scheme Numbers" registry (Section 11.1). The relationship of a scheme number to its scheme-id is as follows:

```
scheme-id = -1 - scheme-number
scheme-number = -1 - scheme-id
```

For example, the scheme-name coap has the (unsigned integer) scheme-number 0 which is represented in a (negative integer) scheme-id -1.

5.1.2. 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 trailing segment, if any, from the base URI's path, which is represented in the equivalent CRI reference discard section as the value 1; similarly "../foo" removes two trailing segments, if any, 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 RFC 3986 [STD66] calls "relative references" (i.e., references that need to undergo a resolution process to obtain a URI) correspond to the CRI reference form that starts with discard. The exception are relative references with an authority (called a "network-path reference" in Section 4.2

of RFC 3986 [STD66]), which discard the entire path of the base CRI. These CRI references never carry a discard section: the value of discard defaults to true.

5.1.3. Examples

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

Figure 3: CRI for coap://198.51.100.1:61616/.well-known/core

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

Figure 4: CRI Reference for /.well-known/core?rt=temperature-c

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

Figure 5: CRI for did:web:alice:bob

5.1.4. Specific Terminology

A CRI reference is considered _well-formed_ if it matches the structure as expressed in Figure 2 in CDDL, with the additional requirement that trailing null values are removed from the array.

A CRI reference is considered a _full_ CRI 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

We refer to this as the _abstract form_, while the _interchange form_ (Figure 2) has either two sections for scheme and authority or one section for discard, but never both of these alternatives.

Each of the sections in the abstract form can be _not set_ ("null"), except for discard, which is always an unsigned integer or true. If scheme and/or authority are non-null, discard is set to true.

When ingesting a CRI reference that is in interchange form, those sections are filled in from interchange form (sections not set are filled with null), and the following steps are performed:

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

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

* If scheme and/or authority are present in interchange 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 interchange 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 reference "").

5.2.1. Error handling and extensibility

It is recommended that specifications that describe the use of CRIs in CBOR-based protocols use the error handling mechanisms outlined in this section. Implementations of this document MUST adhere to these rules unless a containing document overrides them.

When encountering a CRI that is well-formed in terms of CBOR, but that

* is not well-formed as a CRI,

* does not meet the other requirements on CRIs that are not covered by the term "well-formed", or

* uses features not supported by the implementation,

the CRI is treated as "unprocessable".

When encountering an unprocessable CRI, the processor skips the entire CRI top-level array, including any CBOR items contained therein, and continues processing the CBOR items surrounding the unprocessable CRI. (Note: this skipping can be implemented in bounded memory for CRIs that do not use indefinite length encoding, as mandated for CRIs as separate encoded data items in Section 5.1, Paragraph 10.)

The unprocessable CRI is treated as an opaque identifier that is distinct from all processable CRIs, and distinct from all unprocessable CRIs with different CBOR representations. It is up to the implementation whether unprocessable CRIs with identical representations are treated as identical to each other or not. Unprocessable CRIs cannot be dereferenced, and it is an error to query any of their components.

This mechanism ensures that CRI extensions (using originally defined features or later extensions) can be used without extending the compatibility hazard to the containing document. For example, if a collection of possible interaction targets contains several CRIs, some of which use the "no-authority" feature, an application consuming that collection that does not support that feature can still offer the supported interaction targets.

The duty of checking validity is with the recipients that rely on this validity. An intermediary that does not use the detailed information in a CRI (or merely performs reference resolution) MAY pass on a CRI/CRI reference without having fully checked it, relying on the producer having generated a valid CRI/CRI reference. This is true for both Simple CRIs (e.g., checking for valid UTF-8) and for extensions (e.g., checking both for valid UTF-8 and the minimal use of PET elements in the text-or-pet feature as per Section 7.2).

A system that is checking a CRI for some reason but is not its ultimate recipient needs to consider the tension between security requirements and the danger of ossification [RFC9170]: If the system rejects anything that it does not know, it prevents the other components from making use of extensions. If it passes through extensions unknown to it, that might allow semantics pass through that the system should have been designed to filter out.

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 full CRI in the form of an CRI reference:

1. Establish the base CRI of the CRI reference (compare Section 5.1 of RFC 3986 [STD66]) and express it in the form of an abstract (full) 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, set query to empty and fragment to null, and set a true authority to null. If the value of discard is an unsigned integer, remove as many elements from the end of the path array; if it is non-zero, set query to empty and fragment to null.

   Set discard to true in the buffer.

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

5. If query is non-null in the CRI reference, set fragment to null in the buffer. Apart from the path and discard, copy all non-null sections from the CRI reference to the buffer in sequence.

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) [STD66] and Internationalized Resource Identifiers (IRIs) [RFC3987] in constrained environments [RFC7228][I-D.ietf-iotops-7228bis]. 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.

  Note that CRIs are defined to enable implementing conversions from or to URIs analogously to processing URIs into CoAP Options and back, with the exception that item 8 of Section 6.4 of [RFC7252] and item 7 of Section 6.5 of [RFC7252] do not apply to CRI processing. See SP1 in Appendix A for more details.

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 about CRI references, URI references, and IRI references in this section also applies to CRIs, URIs, and IRIs.

6.1. Converting CRI (references) to URI (references)

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 RFC 3986 [STD66].

scheme
: If the CRI reference contains a scheme section, the scheme component of the URI reference consists of the value of that section, if text (scheme-name); or, if a negative integer is given (scheme-id), the lower case scheme name corresponding to the scheme-id as per Section 5.1.1. Otherwise, the scheme component is not set.

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 not set.

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 element that is not in the set of unreserved characters (Section 2.3 of RFC 3986 [STD66]) or "sub-delims" (Section 2.2 of RFC 3986 [STD66]) or a colon (":") 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 not in the set of unreserved characters (Section 2.3 of RFC 3986 [STD66]) or "sub-delims" (Section 2.2 of RFC 3986 [STD66]) MUST be percent-encoded. If there are dots (".") in such elements, the conversion fails (percent-encoding is not able to represent such elements, as normalization would turn the percent-encoding back to the unreserved character that a dot is.)

| Implementations with scheme-specific knowledge MAY convert individual elements by using the ToASCII procedure (Section 4.1 of [RFC3490]) as discussed in more detail in Section 3.1 of [RFC3987]. This should not be done if the next step of conversion is to an IRI as defined in Section 6 (CRI to IRI).
|
|
| // Editor’s note: Some other RFCs reference RFC5890 as the source of
| // ToASCII, since that is the document that replaces RFC3490 and at
| // least mentions ToASCII. Unfortunately, this doesn’t define
| // ToASCII (pointing to RFC 3490 instead), so we consider these
| // references broken. Instead, we reference RFC 3490, which is the
| // document that actually does define ToASCII. RFC 3987 (IRIs)
| // references RFC 3490, too, kind of keeping it alive.

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 RFC 3986 [STD66]).

The inclusion of zone-ids [RFC4007] in URIs has a complex history and currently has no interoperable representation (the previous specification for this, [RFC6874], is now obsoleted by [I-D.ietf-6man-zone-ui]; more background information is available in [I-D.schinazi-httpbis-link-local-uri-bcp]). The CRI specification does not define a conversion from a CRI containing a zone-id to a URI. As keeping a zone-id with an IP address in a URI turned out to be useful while [RFC6874] was in effect, CRIs maintain a position in the grammar to optionally store a zone-id. This can be used by consenting CRI implementations to exchange zone information without being concerned by the lack of a specification at the URI syntax level. The goal is to achieve approximate feature parity with the zone-id support in [I-D.ietf-netmod-rfc6991-bis], which also contains further clarifications on the use of zone-ids with IP addresses.

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 value is 1 and the first element of the path contains a :, the path component is prefixed by "./" (this avoids the first element to appear as supplying a URI scheme; compare path-noscheme in Section 4.2 of RFC 3986 [STD66]).

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 RFC 3986 [STD66]), 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 RFC 3986 [STD66]), 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 RFC 3986 [STD66]), 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 not set.

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 not set.

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. Extending CRIs

The CRI structure described up to this point, without enabling any feature ("scheme-name", "no-authority", "userinfo"), is termed the _Basic CRI_. It should be sufficient for all applications that use the CoAP protocol, as well as most other protocols employing URIs.

With one or more of the three features enabled, we speak of _Simple CRIs_, which cover a larger subset of protocols that employ URIs. To overcome remaining limitations, _Extended Forms_ of CRIs may be

defined to enable further applications. They will generally extend the CRI structure to accommodate more potential values of text components of URIs, such as userinfo, hostnames, paths, queries, and fragments.

Extensions may also be defined to afford a more natural representation of the information in a URI. _Stand-in Items_ (Section 7.1) are one way to provide such representations. For instance, information that needs to be base64-encoded in a URI can be represented in a CRI in its natural form as a byte string instead.

Extensions are or will be necessary to cover two limitations of Simple CRIs:

* Simple CRIs do not support IPvFuture (Section 3.2.2 of RFC 3986 [STD66]). Definition of such an extension probably best waits until a wider use of new IP literal formats is planned.

* More important in practice:

  Simple CRIs do not support URI components that _require_ percent-encoding (Section 2.1 of RFC 3986 [STD66]) 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 Simple CRIs:

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

  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.

  An extension supporting application-level percent-encoded text in CRIs is described in Section 7.2.

Consumers of CRIs will generally notice when an extended form is in use, by finding structures that do not match the CDDL rules given in Figure 2. Future definitions of extended forms need to strive to be distinguishable in their structures from the extended form presented here as well as other future forms.

Extensions to CRIs are not intended to change encoding constraints; e.g., Section 5.1, Paragraph 10 is applicable to extended forms of CRIs as well. This also ensures that recipients of CRIs can deal with unprocessable CRIs as described in Section 5.2.1.

7.1. Extended CRI: Stand-In Items

Application specifications that use CRIs may explicitly enable the use of "stand-in" items (tags or simple values). These are data items used in place of original representation items such as strings or arrays, where the tag or simple value is defined to stand for a data item that can be used in the position of the stand-in item. Examples would be (1) tags such as 21 to 23 (Section 3.4.5.2 of RFC 8949 [STD94]) or 108 (Section 2.1 of [I-D.bormann-cbor-notable-tags]), which stand for text string components but internally employ more compact byte string representations, or (2) reference tags and simple values as defined in [I-D.ietf-cbor-packed].

Application specifications need to be explicit about which stand-in items are allowed; otherwise, inconsistent interpretations at different places in a system can lead to check/use vulnerabilities.

(Note that specifications that define CBOR tags may be employed in CRI extensions without actually using the tags defined there as stand-in tags; e.g., compare the way IP addresses are represented in Basic CRIs with [RFC9164].)

7.2. Extended CRI: Accommodating Percent Encoding (PET)

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 "text-or-pet"

; 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 abovementioned DID URI

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

can now be represented as:

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

(Note that, in CBOR diagnostic notation, single quotes delimit literals for byte strings, double quotes for text strings.)

To yield a valid CRI using the text-or-pet feature, the use of byte strings MUST be minimal. Both the following examples are therefore not valid:

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

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

An algorithm for constructing a valid text-pet-sequence might repeatedly examine the byte sequences in each byte string; if such a sequence stands for an unreserved ASCII character, or constitutes a valid UTF-8 character U+0080, move this character over into a text string by appending it to the end of the preceding text string, prepending it to the start of the following text string, or splitting

the byte string and inserting a new text string with this character, all while preserving the order of the bytes. (Note that the properties of UTF-8 make this a simple linear process; working around the NFC constraint C0 in this way may be more complex.)

> | Unlike the text elements of a path or a query, which through CoAP's heritage are designed to be processable element by element, a text-pet-sequence does not usually produce a semantically meaningful division into array elements. This consequence of the flexibility in delimiters offered in URIs is demonstrated by this example, which structurally singles out the one ':' that is _not_ a delimiter at the application level. Applications specifically designed for using CRIs will generally avoid using the text-or-pet feature. Applications using existing URI structures that require text-pet-sequence elements for their representation typically need to process them byte by byte.

8. Integration into CoAP and ACE

This section discusses ways in which CRIs can be used in the context of the CoAP protocol [RFC7252] and of Authorization for Constrained Environments (ACE), specifically the Authorization Information Format (AIF) [RFC9237].

8.1. Converting Between CoAP CRIs and Sets of CoAP Options

This section provides an analogue to Sections 6.4 and 6.5 of [RFC7252]: Computing a set of CoAP options from a request CRI (Section 8.1.1) and computing a request CRI from a set of COAP options (Section 8.1.2).

This section makes use of the mapping between CRI scheme numbers and URI scheme names shown in Table 2:

CRI scheme number	URI scheme name
0	coap
1	coaps
6	coap+tcp
7	coaps+tcp
24	coap+ws
25	coaps+ws

Table 2: Mapping CRI scheme numbers and URI scheme names

8.1.1. Decomposing a Request CRI into a set of CoAP Options

The steps to parse a request's options from a CRI cri are as follows. These steps either result in zero or more of the Uri-Host, Uri-Port, Uri-Path, and Uri-Query Options being included in the request or they fail.

Where the following speaks of deriving a text-string for a CoAP Option value from a data item in the CRI, the presence of any text-pet-sequence subitem (Section 7.2) in this item fails this algorithm.

1. If cri is not a full CRI, then fail this algorithm.

2. Translate the scheme-id into a URI scheme name as per Section 5.1.1 and Table 2; if a scheme-id that corresponds to a scheme number not in this list is being used, or if a scheme-name is being used, fail this algorithm. Remember the specific variant of CoAP to be used based on this URI scheme name.

3. If the cri's fragment component is non-null, then fail this algorithm.

4. If the host component of cri is a host-name, include a Uri-Host Option and let that option's value be the text string values of the host-name elements joined by dots.

   If the host component of cri is a host-ip, check whether the IP address given represents the request's destination IP address (and the zone-ids of both addresses also match by being absent or by pointing to the same interface). Only if it does not, include a Uri-Host Option, and let that option's value be the text value of the URI representation of the IP address, as derived in Section 6.1, Paragraph 3 (note that zone-ids are never present in Uri-Host Options).

5. If the port subcomponent in a cri is not absent, then let port be that subcomponent's unsigned integer value; otherwise, let port be the default port number for the scheme.

6. If port does not equal the request's destination port, include a Uri-Port Option and let that option's value be port.

7. If the value of the path component of cri is empty or consists of a single empty string, then move to the next step.

   Otherwise, for each element in the path component, include a Uri-Path Option and let that option's value be the text string value of that element.

8. If the value of the query component of cri is non-empty, then, for each element in the query component, include a Uri-Query Option and let that option’s value be the text string value of that element.

8.1.2. Composing a Request CRI from a Set of CoAP Options

The steps to construct a CRI from a request’s options are as follows. These steps either result in a CRI or they fail.

1. Based on the variant of CoAP used in the request, choose a scheme-id as per Section 5.1.1 and table Table 2. Use that as the first value in the resulting CRI array.

2. If the request includes a Uri-Host Option, insert an authority with its value determined as follows: If the value of the Uri-Host Option is a reg-name, split it on any dots in the name and use the resulting text string values as the elements of the host-name array. If the value is an IP-literal or IPv4address, represent the IP address as a byte string of the correct length in host-ip; if a zone-id can be extracted from the request’s destination IP address and if the IP address is ambiguous in the context of the local system, append the zone-id. If the value is none of the three, fail this algorithm.

   If the request does not include a Uri-Host Option, insert an authority with host-ip being the byte string that represents the request’s destination IP address and, if one is present in the request’s destination address, add a zone-id.

3. If the request includes a Uri-Port Option, let port be that option’s value. Otherwise, let port be the request’s destination port. If port is not the default port for the scheme, then insert the integer value of port as the value of port in the authority. Otherwise, elide the port.

4. Insert a path component that contains an array built from the text string values of the Uri-Path Options in the request, or an empty array if no such options are present.

5. Insert a query component that contains an array built from the text string values of the Uri-Query Options in the request, or an empty array if no such options are present.

8.2. CoAP Options for Forward-Proxies

Apart from the above procedures to convert CoAP CRIs to and from sets of CoAP Options, two additional CoAP Options are defined in Section 5.10.2 of [RFC7252] that support requests to forward-proxies:

* Proxy-Uri, and

* its more lightweight variant, Proxy-Scheme

This section defines analogues of these that employ CRIs and the URI Scheme numbering provided by the present specification.

8.2.1. Proxy-CRI

No.	C	U	N	R	Name	Format	Length	Default
TBD235	x	x	-		Proxy-Cri	opaque	1-1023	(none)

Table 3: Proxy-Cri CoAP Option

The Proxy-CRI Option carries an encoded CBOR data item that represents a full CRI. It is used analogously to Proxy-Uri as defined in Section 5.10.2 of [RFC7252]. The Proxy-Cri Option MUST take precedence over any of the Uri-Host, Uri-Port, Uri-Path or Uri-Query options, as well as over any Proxy-Uri Option (each of which MUST NOT be included in a request containing the Proxy-Cri Option).

8.2.2. Proxy-Scheme-Number

No.	C	U	N	R	Name	Format	Length	Default
TBD239	x	x	-		Proxy-Scheme-Number	uint	0-3	(none)

Table 4: Proxy-Scheme-Number CoAP Option

The Proxy-Scheme-Number Option carries a CRI Scheme Number represented as a CoAP unsigned integer. It is used analogously to Proxy-Scheme as defined in Section 5.10.2 of [RFC7252].

The Proxy-Scheme Option MUST NOT be included in a request that also contains the Proxy-Scheme-Number Option; servers MUST reject the request if this is the case.

As per Section 3.2 of [RFC7252], CoAP Options are only defined as one of empty, (text) string, opaque (byte string), or uint (unsigned integer). The Option therefore carries an unsigned integer that represents the CRI scheme-number (which relates to a CRI scheme-id as defined in Section 5.1.1). For instance, the scheme name "coap" has the scheme-number 0 and is represented as an unsigned integer by a zero-length CoAP Option value.

8.3. ACE AIF

The AIF (Authorization Information Format, [RFC9237]) defined by ACE by default uses the local part of a URI to identify a resource for which authorization is indicated. The type and target of this information is an extension point, briefly called _Toid_ (Type of object identifier). Section 11.6 registers "CRI-local-part" as a Toid. Together with _Tperm_, an extension point for a way to indicate individual access rights (permissions), Section 2 of [RFC9237] defines its general Information Model as:

```
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
```

Using the definitions in Figure 2 together with the [RFC9237] default Tperm choice REST-method-set, this information model can be specialized as in:

```
CRI-local-part = [path, ?query]
AIF-CRI = AIF-Generic<CRI-local-part, REST-method-set>
```

9. Implementation Status

(Boilerplate as per Section 2.1 of [RFC7942]:)

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation

and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

A golang implementation of revision -10 of this document is found at: https://github.com/thomas-fossati/href. A Rust implementation is available at https://codeberg.org/chrysn/cri-ref; it is being updated to revision -18 at the time of writing. A python implementation is available as part of https://gitlab.com/chrysn/micrurus but is based on revision -05.

10. 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 RFC 8949 [STD94] for additional security considerations relating to CBOR.

The security considerations discussed in Section 7 of RFC 3986 [STD66] and Section 8 of [RFC3987] for URIs and IRIs also apply to CRIs. The security considerations discussed for URIs in Section 6 of [RFC9237] apply analogously to AIF-CRI Section 8.3.

11. IANA Considerations

// RFC-editor: Please replace all references to Appendix C by a
// reference to the IANA registry.

// RFC Ed.: throughout this section, please replace RFC-XXXX with the
// RFC number of this specification and remove this note.

11.1. CRI Scheme Numbers Registry

This specification defines a new "CRI Scheme Numbers" registry in the "Constrained RESTful Environments (CoRE) Parameters" registry group [IANA.core-parameters], with the policy "Expert Review" (Section 4.5 of RFC 8126 [BCP26]). The objective is to have CRI scheme number values registered for all registered URI schemes (Uniform Resource Identifier (URI) Schemes registry), as well as exceptionally for certain text strings that the Designated Expert considers widely used in constrained applications in place of URI scheme names.

11.1.1. Instructions for the Designated Expert

The expert is instructed to be frugal in the allocation of CRI scheme number values whose scheme-id values (Section 5.1.1) have short representations (1+0 and 1+1 encoding), keeping them in reserve for applications that are likely to enjoy wide use and can make good use of their shortness.

When the expert notices that a registration has been made in the Uniform Resource Identifier (URI) Schemes registry (see also Section 11.2), the expert is requested to initiate a parallel registration in the CRI Scheme Numbers registry. CRI scheme number values in the range between 1000 and 20000 (inclusive) should be assigned unless a shorter representation in CRIs appears desirable.

The expert exceptionally also may make such a registration for text strings that have not been registered in the Uniform Resource Identifier (URI) Schemes registry if and only if the expert considers them to be in wide use in place of URI scheme names in constrained applications. (Note that registrations in the CRI Scheme Numbers registry are oblivious to the details of any URI Schemes registry registration, so if a registration is later made in the URI Schemes registry that uses such a previously unregistered text string as a name, the CRI Scheme Numbers registration simply stays in place, even if the URI Schemes registration happens to be for something different from what the expert had in mind at the time for the CRI Scheme Numbers registration. Also note that the initial registrations in Table 10 in Appendix C already include such registrations for the text strings "mqtt" and "mqtts".)

A registration in the CRI Scheme Numbers registry does not imply that a URI scheme under this name exists or has been registered in the Uniform Resource Identifier (URI) Schemes registry -- it essentially is only providing an integer identifier for an otherwise uninterpreted text string.

Any questions or issues that might interest a wider audience might be raised by the expert on the core-parameters@ietf.org mailing list for a time-limited discussion.

11.1.2. Structure of Entries

Each entry in the registry must include:

CRI scheme number:
: An unsigned integer unique in this registry

URI scheme name:

a text string that would be acceptable for registration as a URI Scheme Name in the Uniform Resource Identifier (URI) Schemes registry

Reference:
a reference to a document, if available, or the registrant

The Reference field can simply be a copy of the reference field for the URI-Scheme registration if that exists. If not, it can contain helpful information (including the name of the registrant) that may be available for the registration, with the expectation that this will be updated if a URI-Scheme registration under that URI scheme name is later made.

11.1.3. Initial Registrations

The initial registrations for the CRI Scheme Numbers registry are provided in Table 10 in Appendix C.

11.2. Update to "Uniform Resource Identifier (URI) Schemes" Registry

RFC 7595 [BCP35] is updated to add the following note in the "Uniform Resource Identifier (URI) Schemes" Registry [IANA.uri-schemes]:

> The CRI Scheme Numbers Registry registers numeric identifiers for what essentially are URI Scheme names. Registrants for the Uniform Resource Identifier (URI) Schemes Registry are requested to make a parallel registration in the CRI Scheme Numbers registry. The number for this registration will be assigned by the Designated Expert for that registry.

11.3. CBOR Diagnostic Notation Application-extension Identifiers Registry

In the "Application-Extension Identifiers" registry in the "CBOR Diagnostic Notation" registry group [IANA.cbor-diagnostic-notation], IANA is requested to register the application-extension identifier cri as described in Table 5 and defined in Appendix B.

Application-extension Identifier	Description	Change Controller	Reference
cri	Constrained Resource Identifier	IETF	RFC-XXXX, Appendix B

Table 5: CBOR Extended Diagnostic Notation (EDN) Application-extension Identifier for CRI

11.4. CBOR Tags Registry

```
// RFC-Editor: This document uses the CPA (code point allocation)
// convention described in [I-D.bormann-cbor-draft-numbers].  For
// each usage of the term "CPA", please remove the prefix "CPA" from
// the indicated value and replace the residue with the value
// assigned by IANA; perform an analogous substitution for all other
// occurrences of the prefix "CPA" in the document.  Finally, please
// remove this note.
```

In the "CBOR Tags" registry [IANA.cbor-tags], IANA is requested to assign the tags in Table 6 from the "specification required" space (suggested assignment: 99), with the present document as the specification reference.

Tag	Data Item	Semantics	Reference
CPA99	array	CRI Reference	RFC-XXXX

Table 6: Values for Tags

11.5. CoAP Option Numbers Registry

```
// RFC-Editor: For each usage of the term "TBD", please remove the
// prefix "TBD" from the indicated value and replace the residue with
// the value actually assigned by IANA; perform an analogous
// substitution for all other occurrences of the prefix "TBD" in the
// document.  Finally, please remove this note.
```

In the "CoAP Option Numbers" registry in the "CoRE Parameters" registry group [IANA.core-parameters], IANA is requested to register the CoAP Option Numbers as described in Table 7 and defined in Section 8.2.

No.	Name	Reference
TBD235	Proxy-Cri	RFC-XXXX
TBD239	Proxy-Scheme-Number	RFC-XXXX

Table 7: New CoAP Option Numbers

11.6. Media-Type subparameters for ACE AIF

In the "Sub-Parameter Registry for application/aif+cbor and application/aif+json" in the "Media Type Sub-Parameter Registries" registry group [IANA.media-type-sub-parameters], IANA is requested to register:

Parameter	Name	Description/ Specification	Reference
Toid	CRI-local-part	local-part of CRI	Section 8.3 of RFC-XXXX

Table 8: ACE AIF Toid for CRI

11.7. Content-Format for CRI in AIF

IANA is requested to register a Content-Format identifier in the "CoAP Content-Formats" registry (range 256-999), within the "Constrained RESTful Environments (CoRE) Parameters" registry group [IANA.core-parameters], as follows:

Content Type	Content Coding	ID	Reference
application/ aif+cbor;Toid=CRI-local-part	-	TBD	RFC-XXXX

Table 9: Content-Format for ACE AIF with CRI-local-part Toid

12. References

12.1. Normative References

[BCP14] Best Current Practice 14,
<https://www.rfc-editor.org/info/bcp14>.
At the time of writing, this BCP comprises the following:

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>.

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>.

[BCP26] Best Current Practice 26,
<https://www.rfc-editor.org/info/bcp26>.
At the time of writing, this BCP comprises the following:

Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[BCP35] Best Current Practice 35,
<https://www.rfc-editor.org/info/bcp35>.
At the time of writing, this BCP comprises the following:

Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/info/rfc7595>.

[I-D.ietf-netmod-rfc6991-bis]
Schnwlder, J., "Common YANG Data Types", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc6991-bis-18, 23 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc6991-bis-18>.

[IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <https://www.iana.org/assignments/cbor-tags>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

[IANA.media-type-sub-parameters]
IANA, "Media Type Sub-Parameter Registries", <https://www.iana.org/assignments/media-type-sub-parameters>.

[IANA.uri-schemes]
IANA, "Uniform Resource Identifier (URI) Schemes", <https://www.iana.org/assignments/uri-schemes>.

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

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <https://www.rfc-editor.org/rfc/rfc4007>.

[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/rfc/rfc8610>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/rfc/rfc9165>.

[RFC9237] Bormann, C., "An Authorization Information Format (AIF) for Authentication and Authorization for Constrained Environments (ACE)", RFC 9237, DOI 10.17487/RFC9237, August 2022, <https://www.rfc-editor.org/rfc/rfc9237>.

[STD63] Internet Standard 63, <https://www.rfc-editor.org/info/std63>.
At the time of writing, this STD comprises the following:

Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[STD66] Internet Standard 66, <https://www.rfc-editor.org/info/std66>.
At the time of writing, this STD comprises the following:

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>.

[STD94] Internet Standard 94, <https://www.rfc-editor.org/info/std94>. At the time of writing, this STD comprises the following:

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/>. RFC Editor: please replace with version current at publication (probably 17.0.0) and check whether D80, D120, and D92 are still pointing to the same definitions as in 13.0.0.

12.2. Informative References

[BCP190] Best Current Practice 190, <https://www.rfc-editor.org/info/bcp190>. At the time of writing, this BCP comprises the following:

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

[I-D.bormann-cbor-notable-tags] Bormann, C., "Notable CBOR Tags", Work in Progress, Internet-Draft, draft-bormann-cbor-notable-tags-12, 12 February 2025, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-notable-tags-12>.

[I-D.ietf-6man-zone-ui] Carpenter, B. E. and R. M. Hinden, "Entering IPv6 Zone Identifiers in User Interfaces", Work in Progress, Internet-Draft, draft-ietf-6man-zone-ui-10, 19 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-6man-zone-ui-10>.

[I-D.ietf-cbor-edn-literals]
Bormann, C., "CBOR Extended Diagnostic Notation (EDN)", Work in Progress, Internet-Draft, draft-ietf-cbor-edn-literals-18, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-18>.

[I-D.ietf-cbor-packed]
Bormann, C. and M. Gtschow, "Packed CBOR", Work in Progress, Internet-Draft, draft-ietf-cbor-packed-16, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-packed-16>.

[I-D.ietf-iotops-7228bis]
Bormann, C., Ersue, M., Kernen, A., and C. Gomez, "Terminology for Constrained-Node Networks", Work in Progress, Internet-Draft, draft-ietf-iotops-7228bis-02, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-iotops-7228bis-02>.

[I-D.schinazi-httpbis-link-local-uri-bcp]
Schinazi, D., "Best Practices for Link-Local Connectivity in URI-Based Protocols", Work in Progress, Internet-Draft, draft-schinazi-httpbis-link-local-uri-bcp-03, 22 February 2024, <https://datatracker.ietf.org/doc/html/draft-schinazi-httpbis-link-local-uri-bcp-03>.

[MNU] Bormann, C., "Modern Network Unicode", Work in Progress, Internet-Draft, draft-bormann-dispatch-modern-network-unicode-06, 2 March 2025, <https://datatracker.ietf.org/doc/html/draft-bormann-dispatch-modern-network-unicode-06>.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, DOI 10.17487/RFC3490, March 2003, <https://www.rfc-editor.org/rfc/rfc3490>.

[RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, <https://www.rfc-editor.org/rfc/rfc4180>.

[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/rfc/rfc6874>.

[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/rfc/rfc7228>.

[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/rfc/rfc7252>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

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

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

[RFC9164] Richardson, M. and C. Bormann, "Concise Binary Object Representation (CBOR) Tags for IPv4 and IPv6 Addresses and Prefixes", RFC 9164, DOI 10.17487/RFC9164, December 2021, <https://www.rfc-editor.org/rfc/rfc9164>.

[RFC9170] Thomson, M. and T. Pauly, "Long-Term Viability of Protocol Extension Mechanisms", RFC 9170, DOI 10.17487/RFC9170, December 2021, <https://www.rfc-editor.org/rfc/rfc9170>.

[STD97] Internet Standard 97, <https://www.rfc-editor.org/info/std97>. At the time of writing, this STD comprises the following:

Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[W3C.REC-html52-20171214] Danilo, A., Ed., Eicholz, A., Ed., Moon, S., Ed., Faulkner, S., Ed., and T. Leithead, Ed., "HTML 5.2", W3C REC REC-html52-20171214, W3C 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 [STD66], s://x is distinct from s://x/ -- i.e., a URI with an empty path ([] in CRI) is different from one with a lone empty path segment (""]). However, in HTTP and 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. Similarly, after discussing HTTP semantics, Section 6.2.3 of RFC 3986 [STD66] 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 leading 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.

  These URIs can be expressed when using the text-or-pet feature.

* 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 7.2) 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 (note that . is doubly special because it is not a reserved character in [STD66] and therefore any percent-encoding would be normalized away).

Thus, the following examples _are_ convertible to basic CRIs without the text-or-pet feature:

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 explicitly cater for unencoded ":" in userinfo (without needing the text-or-pet feature). (We opted for including this syntactic feature instead of disabling it as a mechanism against potential uses of colons for the deprecated inclusion of unencrypted secrets.)

Appendix B. CBOR Extended Diagnostic Notation (EDN): The "cri" Extension

[I-D.ietf-cbor-edn-literals] more rigorously defines and further extends the CBOR Extended Diagnostic Notation (EDN), as originally introduced in Section 8 of RFC 8949 [STD94] and extended in Appendix G of [RFC8610]. Among others, it provides an extension point for "application-extension identifiers" that can be used to notate CBOR data items in application-specific ways.

The present document defines and registers (Section 11.3) the application-extension identifier "cri", which can be used to notate an EDN literal for a CRI reference as defined in this document.

The text of the literal is a URI Reference as per [STD66] or an IRI Reference as per [RFC3987].

The value of the literal is a CRI reference that can be converted to the text of the literal using the procedure of Section 6.1. Note that there may be more than one CRI reference that can be converted to the URI/IRI reference given; implementations are expected to favor the simplest variant available and make non-surprising choices otherwise. In the all-upper-case variant of the app-prefix, the value is enclosed in a tag number CPA99.

```
// RFC-Editor: This document uses the CPA (code point allocation)
// convention described in [I-D.bormann-cbor-draft-numbers]. For
// each usage of the term "CPA", please remove the prefix "CPA" from
// the indicated value and replace the residue with the value
// assigned by IANA; perform an analogous substitution for all other
// occurrences of the prefix "CPA" in the document. Finally, please
// remove this note.
```

As an example, the CBOR diagnostic notation

```
cri'https://example.com/bottarga/shaved'
CRI'https://example.com/bottarga/shaved'
```

is equivalent to

```
[-4, ["example", "com"], ["bottarga", "shaved"]]
CPA99([-4, ["example", "com"], ["bottarga", "shaved"]])
```

See Appendix B.1 for an ABNF definition for the content of cri literals.

B.1. cri: ABNF Definition of URI Representation of a CRI

It can be expected that implementations of the application-extension identifier "cri" will make use of platform-provided URI implementations, which will include a URI parser.

In case such a URI parser is not available or inconvenient to integrate, a grammar of the content of cri literals is provided by the ABNF for URI-reference in Section 4.1 of RFC 3986 [STD66] with certain re-arrangements taken from Figure 5 of [I-D.ietf-cbor-edn-literals]; these are reproduced in Figure 6. If the content is not ASCII only (i.e., for IRIs), first apply Section 3.1 of [RFC3987] and apply this grammar to the result.

```
   app-string-cri = URI-reference
   ; ABNF from RFC 3986:

   URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

   hier-part     = "//" authority path-abempty
                    / path-absolute
                    / path-rootless
                    / path-empty

   URI-reference = URI / relative-ref

   absolute-URI  = scheme ":" hier-part [ "?" query ]

   relative-ref  = relative-part [ "?" query ] [ "#" fragment ]

   relative-part = "//" authority path-abempty
                    / path-absolute
                    / path-noscheme
                    / path-empty

   scheme        = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

   authority     = [ userinfo "@" ] host [ ":" port ]
   userinfo      = *( unreserved / pct-encoded / sub-delims / ":" )
   host          = IP-literal / IPv4address / reg-name
   port          = *DIGIT

   IP-literal    = "[" ( IPv6address / IPvFuture  ) "]"

   IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

   ; Use IPv6address, h16, ls32, IPv4adress, dec-octet as re-arranged
   ; for PEG Compatibility in Figure 5 of [I-D.ietf-cbor-edn-literals]:

   IPv6address   =                            6( h16 ":" ) ls32
                 /                       "::" 5( h16 ":" ) ls32
                 / [ h16               ] "::" 4( h16 ":" ) ls32
                 / [ h16 *1( ":" h16 ) ] "::" 3( h16 ":" ) ls32
                 / [ h16 *2( ":" h16 ) ] "::" 2( h16 ":" ) ls32
                 / [ h16 *3( ":" h16 ) ] "::"    h16 ":"   ls32
                 / [ h16 *4( ":" h16 ) ] "::"              ls32
                 / [ h16 *5( ":" h16 ) ] "::"              h16
                 / [ h16 *6( ":" h16 ) ] "::"

   h16           = 1*4HEXDIG
   ls32          = ( h16 ":" h16 ) / IPv4address
   IPv4address   = dec-octet "." dec-octet "." dec-octet "." dec-octet
```

```
   dec-octet     = "25" %x30-35         ; 250-255
                 / "2" %x30-34 DIGIT    ; 200-249
                 / "1" 2DIGIT           ; 100-199
                 / %x31-39 DIGIT        ; 10-99
                 / DIGIT                ; 0-9
   ALPHA         = %x41-5a / %x61-7a
   DIGIT         = %x30-39
   HEXDIG        = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
   ; case insensitive matching, i.e., including lower case

   reg-name      = *( unreserved / pct-encoded / sub-delims )

   path          = path-abempty    ; begins with "/" or is empty
                    / path-absolute   ; begins with "/" but not "//"
                    / path-noscheme   ; begins with a non-colon segment
                    / path-rootless   ; begins with a segment
                    / path-empty      ; zero characters

   path-abempty  = *( "/" segment )
   path-absolute = "/" [ segment-nz *( "/" segment ) ]
   path-noscheme = segment-nz-nc *( "/" segment )
   path-rootless = segment-nz *( "/" segment )
   path-empty    = 0<pchar>

   segment       = *pchar
   segment-nz    = 1*pchar
   segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
                    ; non-zero-length segment without any colon ":"

   pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"

   query         = *( pchar / "/" / "?" )

   fragment      = *( pchar / "/" / "?" )

   pct-encoded   = "%" HEXDIG HEXDIG

   unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
   reserved      = gen-delims / sub-delims
   gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"
   sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
                    / "*" / "+" / "," / ";" / "="
```

Figure 6: ABNF Definition of URI Representation of a CRI

Appendix C. Mapping Scheme Numbers to Scheme Names

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

// RFC Ed.: throughout this section, please replace RFC-XXXX with the
// RFC number of this specification and remove this note.

Table 10 defines the initial mapping from CRI scheme numbers to URI scheme names.

CRI scheme number	URI scheme name	Reference
0	coap	[RFC-XXXX]
1	coaps	[RFC-XXXX]
2	http	[RFC-XXXX]
3	https	[RFC-XXXX]
4	urn	[RFC-XXXX]
5	did	[RFC-XXXX]
6	coap+tcp	[RFC-XXXX]
7	coaps+tcp	[RFC-XXXX]
24	coap+ws	[RFC-XXXX]
25	coaps+ws	[RFC-XXXX]
1059	ms-gamingoverlay	[RFC-XXXX]
1165	snmp	[RFC-XXXX]
1220	cast	[RFC-XXXX]
1242	openid	[RFC-XXXX]
1273	hs20	[RFC-XXXX]
1319	z39.50	[RFC-XXXX]
1328	dweb	[RFC-XXXX]
1466	psyc	[RFC-XXXX]
1528	ms-people	[RFC-XXXX]
1562	ms-personacard	[RFC-XXXX]
1578	jar	[RFC-XXXX]
1658	wpid	[RFC-XXXX]
1762	payment	[RFC-XXXX]
1895	news	[RFC-XXXX]
1905	irc6	[RFC-XXXX]
1926	turns	[RFC-XXXX]
1946	data	[RFC-XXXX]
1982	ens	[RFC-XXXX]
2154	things	[RFC-XXXX]
2284	resource	[RFC-XXXX]
2326	skype	[RFC-XXXX]
2406	videotex	[RFC-XXXX]
2442	dpp	[RFC-XXXX]

2747	upt	[RFC-XXXX]
2754	platform	[RFC-XXXX]
2790	ed2k	[RFC-XXXX]
2796	taler	[RFC-XXXX]
2806	fm	[RFC-XXXX]
2945	ms-newsandinterests	[RFC-XXXX]
3005	xmlrpc.beep	[RFC-XXXX]
3018	ark	[RFC-XXXX]
3119	wss	[RFC-XXXX]
3143	tel	[RFC-XXXX]
3255	vscode-insiders	[RFC-XXXX]
3342	geo	[RFC-XXXX]
3348	rtmfp	[RFC-XXXX]
3358	mtqp	[RFC-XXXX]
3365	filesystem	[RFC-XXXX]
3375	teapots	[RFC-XXXX]
3503	proxy	[RFC-XXXX]
3524	sms	[RFC-XXXX]
3634	jms	[RFC-XXXX]
3646	mid	[RFC-XXXX]
3690	ms-calculator	[RFC-XXXX]
3775	gitoid	[RFC-XXXX]
3783	calculator	[RFC-XXXX]
3786	about	[RFC-XXXX]
3795	facetime	[RFC-XXXX]
3818	ari	[RFC-XXXX]
3837	ymsgr	[RFC-XXXX]
3886	dict	[RFC-XXXX]
3906	ldaps	[RFC-XXXX]
3920	rtmp	[RFC-XXXX]
3959	ms-settings-proximity	[RFC-XXXX]
4053	fax	[RFC-XXXX]
4102	ms-drive-to	[RFC-XXXX]
4153	res	[RFC-XXXX]
4183	webcal	[RFC-XXXX]
4193	embedded	[RFC-XXXX]
4315	xftp	[RFC-XXXX]
4327	browserext	[RFC-XXXX]
4355	session	[RFC-XXXX]
4373	dav	[RFC-XXXX]
4419	ipps	[RFC-XXXX]
4515	uuid-in-package	[RFC-XXXX]
4549	dhttp	[RFC-XXXX]
4559	web3	[RFC-XXXX]
4590	iris.lwz	[RFC-XXXX]
4598	diaspora	[RFC-XXXX]
4613	ms-widgets	[RFC-XXXX]
4619	rtsps	[RFC-XXXX]

4674	beshare	[RFC-XXXX]
4709	gtalk	[RFC-XXXX]
4714	hxxps	[RFC-XXXX]
4747	xrcp	[RFC-XXXX]
4882	sgn	[RFC-XXXX]
4929	eid	[RFC-XXXX]
4951	submit	[RFC-XXXX]
5099	ar	[RFC-XXXX]
5109	ms-settings-airplanemode	[RFC-XXXX]
5134	steam	[RFC-XXXX]
5150	adt	[RFC-XXXX]
5152	ms-appinstaller	[RFC-XXXX]
5188	bb	[RFC-XXXX]
5217	udp	[RFC-XXXX]
5296	example	[RFC-XXXX]
5347	ms-remotedesktop	[RFC-XXXX]
5410	ms-sttoverlay	[RFC-XXXX]
5425	irc	[RFC-XXXX]
5472	sieve	[RFC-XXXX]
5477	machineProvisioningProgressReporter	[RFC-XXXX]
5480	lvlt	[RFC-XXXX]
5492	sftp	[RFC-XXXX]
5536	ms-excel	[RFC-XXXX]
5557	dlna-playcontainer	[RFC-XXXX]
5705	go	[RFC-XXXX]
5717	fido	[RFC-XXXX]
5728	chrome	[RFC-XXXX]
5823	shc	[RFC-XXXX]
5825	swidpath	[RFC-XXXX]
5883	microsoft.windows.camera.picker	[RFC-XXXX]
5990	crid	[RFC-XXXX]
6007	at	[RFC-XXXX]
6024	hcp	[RFC-XXXX]
6030	content-type	[RFC-XXXX]
6109	jabber	[RFC-XXXX]
6144	dlna-playsingle	[RFC-XXXX]
6189	ms-spd	[RFC-XXXX]
6341	opaquelocktoken	[RFC-XXXX]
6349	soldat	[RFC-XXXX]
6380	z39.50s	[RFC-XXXX]
6388	ms-media-stream-id	[RFC-XXXX]
6411	ms-mixedrealitycapture	[RFC-XXXX]
6462	quic-transport	[RFC-XXXX]
6503	ham	[RFC-XXXX]
6516	nfs	[RFC-XXXX]
6609	ut2004	[RFC-XXXX]
6632	hydrazone	[RFC-XXXX]
6634	adiumxtra	[RFC-XXXX]

6651	tip	[RFC-XXXX]
6658	lpa	[RFC-XXXX]
6730	cstr	[RFC-XXXX]
6755	ms-settings-screenrotation	[RFC-XXXX]
6774	dab	[RFC-XXXX]
6792	ms-inputapp	[RFC-XXXX]
6808	moz	[RFC-XXXX]
6840	acd	[RFC-XXXX]
6863	ms-access	[RFC-XXXX]
6883	im	[RFC-XXXX]
6903	pttp	[RFC-XXXX]
6924	teamspeak	[RFC-XXXX]
6992	payto	[RFC-XXXX]
7074	secret-token	[RFC-XXXX]
7126	iax	[RFC-XXXX]
7225	isostore	[RFC-XXXX]
7226	bitcoincash	[RFC-XXXX]
7285	smb	[RFC-XXXX]
7364	appdata	[RFC-XXXX]
7456	dtn	[RFC-XXXX]
7520	feed	[RFC-XXXX]
7667	ssh	[RFC-XXXX]
7743	ms-transit-to	[RFC-XXXX]
7809	ms-help	[RFC-XXXX]
7812	vscode	[RFC-XXXX]
7856	apt	[RFC-XXXX]
7868	ms-settings-notifications	[RFC-XXXX]
7874	shttp (OBSOLETE)	[RFC-XXXX]
7913	ethereum	[RFC-XXXX]
7923	tv	[RFC-XXXX]
7942	microsoft.windows.camera.multipicker	[RFC-XXXX]
8041	msnim	[RFC-XXXX]
8085	ms-remotedesktop-launch	[RFC-XXXX]
8093	spiffe	[RFC-XXXX]
8099	redis	[RFC-XXXX]
8159	z39.50r	[RFC-XXXX]
8251	brid	[RFC-XXXX]
8300	tftp	[RFC-XXXX]
8387	content	[RFC-XXXX]
8454	wais	[RFC-XXXX]
8506	view-source	[RFC-XXXX]
8519	soap.beep	[RFC-XXXX]
8577	attachment	[RFC-XXXX]
8601	gopher	[RFC-XXXX]
8687	ircs	[RFC-XXXX]
8713	callto	[RFC-XXXX]
8765	bolo	[RFC-XXXX]
8766	notes	[RFC-XXXX]

8775	ipn	[RFC-XXXX]
8830	ms-infopath	[RFC-XXXX]
9075	ms-settings	[RFC-XXXX]
9136	ms-useractivityset	[RFC-XXXX]
9154	modem	[RFC-XXXX]
9186	bitcoin	[RFC-XXXX]
9198	ms-settings-privacy	[RFC-XXXX]
9204	cap	[RFC-XXXX]
9278	com-eventbrite-attendee	[RFC-XXXX]
9312	pkcs11	[RFC-XXXX]
9318	ipp	[RFC-XXXX]
9338	rediss	[RFC-XXXX]
9444	grd	[RFC-XXXX]
9453	ms-screensketch	[RFC-XXXX]
9487	matrix	[RFC-XXXX]
9520	xcon-userid	[RFC-XXXX]
9535	sips	[RFC-XXXX]
9544	simpleledger	[RFC-XXXX]
9585	mvn	[RFC-XXXX]
9770	keyparc	[RFC-XXXX]
9805	magnet	[RFC-XXXX]
9816	vsls	[RFC-XXXX]
9859	drm	[RFC-XXXX]
9875	hcap	[RFC-XXXX]
9910	wtai	[RFC-XXXX]
9965	num	[RFC-XXXX]
9981	ms-settings-language	[RFC-XXXX]
10119	imap	[RFC-XXXX]
10147	query	[RFC-XXXX]
10176	ves	[RFC-XXXX]
10183	ms-recall	[RFC-XXXX]
10196	acr	[RFC-XXXX]
10225	barion	[RFC-XXXX]
10229	acct	[RFC-XXXX]
10238	palm	[RFC-XXXX]
10241	ocf	[RFC-XXXX]
10247	lid	[RFC-XXXX]
10317	h323	[RFC-XXXX]
10327	aim	[RFC-XXXX]
10333	turn	[RFC-XXXX]
10361	ms-stickers	[RFC-XXXX]
10373	ms-settings-location	[RFC-XXXX]
10380	dvb	[RFC-XXXX]
10467	xcon	[RFC-XXXX]
10518	ms-screenclip	[RFC-XXXX]
10551	pop	[RFC-XXXX]
10583	dat	[RFC-XXXX]
10591	ms-settings-nfctransactions	[RFC-XXXX]

10640	ms-settings-cloudstorage	[RFC-XXXX]
10687	afs	[RFC-XXXX]
10740	mqtt	[RFC-XXXX]
10744	gizmoproject	[RFC-XXXX]
10831	amss	[RFC-XXXX]
10868	mailserver	[RFC-XXXX]
10926	ni	[RFC-XXXX]
10995	telnet	[RFC-XXXX]
11055	gg	[RFC-XXXX]
11060	blob	[RFC-XXXX]
11072	ms-settings-emailandaccounts	[RFC-XXXX]
11130	ms-project	[RFC-XXXX]
11255	xri	[RFC-XXXX]
11315	msrp	[RFC-XXXX]
11351	ms-settings-connectabledevices	[RFC-XXXX]
11393	cabal	[RFC-XXXX]
11428	nih	[RFC-XXXX]
11467	ms-whiteboard	[RFC-XXXX]
11533	smp	[RFC-XXXX]
11537	vnc	[RFC-XXXX]
11583	graph	[RFC-XXXX]
11645	dvx	[RFC-XXXX]
11718	lorawan	[RFC-XXXX]
11742	lastfm	[RFC-XXXX]
11799	w3	[RFC-XXXX]
11804	mumble	[RFC-XXXX]
11820	thzp	[RFC-XXXX]
11824	feedready	[RFC-XXXX]
11857	microsoft.windows.camera	[RFC-XXXX]
11892	wcr	[RFC-XXXX]
11945	ms-mobileplans	[RFC-XXXX]
11950	ms-settings-lock	[RFC-XXXX]
11962	ws	[RFC-XXXX]
11999	rtspu	[RFC-XXXX]
12029	ms-settings-displays-topology	[RFC-XXXX]
12052	bluetooth	[RFC-XXXX]
12068	file	[RFC-XXXX]
12102	mailto	[RFC-XXXX]
12174	ms-launchremotedesktop	[RFC-XXXX]
12237	ilstring	[RFC-XXXX]
12242	cvs	[RFC-XXXX]
12337	mms	[RFC-XXXX]
12400	ssb	[RFC-XXXX]
12422	iris.xpc	[RFC-XXXX]
12458	starknet	[RFC-XXXX]
12478	qb	[RFC-XXXX]
12493	mss	[RFC-XXXX]
12502	ventrilo	[RFC-XXXX]

12525	ms-lockscreencomponent-config	[RFC-XXXX]
12566	icap	[RFC-XXXX]
12569	mupdate	[RFC-XXXX]
12599	paparazzi	[RFC-XXXX]
12603	ms-widgetboard	[RFC-XXXX]
12634	fish	[RFC-XXXX]
12644	sip	[RFC-XXXX]
12699	mt	[RFC-XXXX]
12705	acap	[RFC-XXXX]
12718	casts	[RFC-XXXX]
12726	reload	[RFC-XXXX]
12732	spotify	[RFC-XXXX]
12806	fuchsia-pkg	[RFC-XXXX]
12823	ms-gamebarservices	[RFC-XXXX]
12876	hyper	[RFC-XXXX]
12932	dns	[RFC-XXXX]
13014	doi	[RFC-XXXX]
13026	ms-settings-power	[RFC-XXXX]
13068	git	[RFC-XXXX]
13094	openpgp4fpr	[RFC-XXXX]
13098	ms-secondary-screen-controller	[RFC-XXXX]
13228	mvrps	[RFC-XXXX]
13285	snews	[RFC-XXXX]
13340	smtp	[RFC-XXXX]
13348	pack	[RFC-XXXX]
13362	teliaeid	[RFC-XXXX]
13372	mongodb	[RFC-XXXX]
13404	afp	[RFC-XXXX]
13440	msrps	[RFC-XXXX]
13442	ldap	[RFC-XXXX]
13451	mvrp	[RFC-XXXX]
13499	nntp	[RFC-XXXX]
13608	onenote	[RFC-XXXX]
13650	sarif	[RFC-XXXX]
13680	elsi	[RFC-XXXX]
13829	otpauth	[RFC-XXXX]
13846	info	[RFC-XXXX]
13862	aaa	[RFC-XXXX]
13923	svn	[RFC-XXXX]
13986	iris	[RFC-XXXX]
14010	lbry	[RFC-XXXX]
14034	ms-search	[RFC-XXXX]
14090	ms-browser-extension	[RFC-XXXX]
14153	maps	[RFC-XXXX]
14162	swid	[RFC-XXXX]
14168	ms-officeapp	[RFC-XXXX]
14180	ms-settings-bluetooth	[RFC-XXXX]
14310	ms-enrollment	[RFC-XXXX]

14347	dntp	[RFC-XXXX]
14364	ms-walk-to	[RFC-XXXX]
14366	ms-getoffice	[RFC-XXXX]
14367	thismessage	[RFC-XXXX]
14460	message	[RFC-XXXX]
14477	prospero	[RFC-XXXX]
14526	aaas	[RFC-XXXX]
14595	market	[RFC-XXXX]
14627	stun	[RFC-XXXX]
14667	chrome-extension	[RFC-XXXX]
14830	itms	[RFC-XXXX]
14860	ms-whiteboard-cmd	[RFC-XXXX]
14867	wifi	[RFC-XXXX]
14868	icon	[RFC-XXXX]
14878	ftp	[RFC-XXXX]
14901	stuns	[RFC-XXXX]
14906	mqtts	[RFC-XXXX]
14936	ms-settings-workplace	[RFC-XXXX]
14962	tn3270	[RFC-XXXX]
14972	pres	[RFC-XXXX]
14982	p1	[RFC-XXXX]
15026	teapot	[RFC-XXXX]
15061	android	[RFC-XXXX]
15118	simplex	[RFC-XXXX]
15163	ms-visio	[RFC-XXXX]
15202	cid	[RFC-XXXX]
15206	unreal	[RFC-XXXX]
15230	tool	[RFC-XXXX]
15254	ms-secondary-screen-setup	[RFC-XXXX]
15267	rtsp	[RFC-XXXX]
15306	xfire	[RFC-XXXX]
15358	xmpp	[RFC-XXXX]
15361	ms-settings-cellular	[RFC-XXXX]
15461	shelter	[RFC-XXXX]
15579	v-event	[RFC-XXXX]
15639	iris.beep	[RFC-XXXX]
15641	wyciwyg	[RFC-XXXX]
15645	ms-meetnow	[RFC-XXXX]
15679	ms-search-repair	[RFC-XXXX]
15773	ms-settings-camera	[RFC-XXXX]
15776	ms-virtualtouchpad	[RFC-XXXX]
15805	xmlrpc.beeps	[RFC-XXXX]
15972	ipfs	[RFC-XXXX]
15994	ms-settings-wifi	[RFC-XXXX]
16051	aw	[RFC-XXXX]
16069	first-run-pen-experience	[RFC-XXXX]
16079	oid	[RFC-XXXX]
16134	iris.xpcs	[RFC-XXXX]

16138	drop	[RFC-XXXX]
16194	ms-publisher	[RFC-XXXX]
16281	leaptofrogans	[RFC-XXXX]
16292	rmi	[RFC-XXXX]
16300	soap.beeps	[RFC-XXXX]
16377	tag	[RFC-XXXX]
16585	ms-word	[RFC-XXXX]
16632	onenote-cmd	[RFC-XXXX]
16645	ms-powerpoint	[RFC-XXXX]
16728	hxxp	[RFC-XXXX]
16729	secondlife	[RFC-XXXX]
16884	rsync	[RFC-XXXX]
16918	vemmi	[RFC-XXXX]
16933	ipns	[RFC-XXXX]
17039	swh	[RFC-XXXX]
17068	pwid	[RFC-XXXX]
17097	dtmi	[RFC-XXXX]
17134	dis	[RFC-XXXX]
17170	iotdisco	[RFC-XXXX]
17175	ms-restoretabcompanion	[RFC-XXXX]
17264	service	[RFC-XXXX]
17315	finger	[RFC-XXXX]
17361	web+ap	[RFC-XXXX]
17381	ms-eyecontrolspeech	[RFC-XXXX]

Table 10: Mapping Scheme Numbers to Scheme Names

The assignments from this table can be extracted from the XML form of this document (when stored in a file "this.xml") into CSV form [RFC4180] using this short Ruby program:

```
require 'rexml/document'; include REXML
XPath.each(Document.new(File.read("this.xml")),"/rfc/back//tr") {|r|
  puts XPath.each(r,"td").map{|d|d.text()}[0..1].join(",")}
```

Appendix D. Change Log

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

Changes from -16 to -17

(Provisional integration of active PRs, please see github.)

Changes from -15 to -16

* Add note that CRI Scheme Number registrations are oblivious of the actual URI Scheme registrations (if any).

* Add information about how this RFC updates RFC7595 to abstract and introduction.

Changes from -14 to -15

* Make scheme numbers unsigned and map them to negative numbers used as scheme-id values

Changes from -09 to -14

* Editorial changes; move some examples to Appendix A, break up railroad diagram; mention commonalities with (and tiny difference from) CoAP Options; mention failure of percent-encoding for dots in host-name components

* Explicitly mention invalid case in Section 2.3, Paragraph 5, Item 3 (rootless CRIs without authority that do not have a path component)

* Generalize Section 7, discuss PET (percent-encoded text) extension in more detail

* Add registry of URI scheme numbers (Appendix C, Section 11)

* Add user information to the authority ("userinfo" feature)

* Figure 2: Use separate rule for CRI, allow [] for query in CRI Reference; generalize scheme numbers, add userinfo; add list of additional requirements in prose (Section 5.1, Paragraph 7)

* Discuss Error handling and extensibility (Section 5.2.1)

* Conversion to URI: Handle : in first pathname component of a CRI-Reference (Section 6.1, Paragraph 6.2.1)

* Add Christian Amsss as contributor

* Add CBOR EDN application-extension "cri" (see Appendix B and Section 11.3).

* Add Section on CoAP integration (and new CoAP Options Proxy-Cri and Proxy-Scheme-Number).

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 [STD66]). 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 [RFC4007] 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 Amsss, Thomas Fossati, Ari Kernen, 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

Christian Amsss
Hollandstr. 12/4
1020 Vienna
Austria
Email: christian@amsuess.com

Authors' Addresses

Carsten Bormann (editor)
Universitt 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