CoRE
Internet-Draft
Updates: 7252 (if approved)
Intended status: Standards Track
Expires: 20 September 2026

C. Amsüss
M. Richardson
Sandelman Software Works

19 March 2026

URI-Path abbreviation in CoAP
draft-ietf-core-uri-path-abbrev-04

Abstract

Applications built on CoAP face a conflict between the technical need for short message sizes and the interoperability requirement of following BCP190 and thus using (relatively verbose) well-known URI paths. This document introduces a CoAP option that allows expressing well-known URI paths in as little as two bytes.

Negotiating the use of this option between client and server revealed a subtle flaw in RFC7252. This document updates RFC7252 to rectify it, thus making the extension point of critical options more useful.

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-uri-path-abbrev/.

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/uri-path-abbrev.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

When building application components on CoAP ([RFC7252]), sending small messages is a general goal in the ecosystem (i.e., constrained environments, where data rates are limited and large packets can lead to packet loss, see [RFC7228]). While CoAP can operate with a wide range of URIs, short path names are therefore favored.

Those short path names need to be discovered, and [RFC7252] and [RFC6690] provide mechanisms for that. Applications that can not discover such paths because they precede a discovery step, such as the discovery itself, setting up a security context ([RFC9528]) or establishing an initial identity ([RFC9148]) can not rely on discovered short paths, and need to use well-known paths. The best practice established in [BCP190] requires applications to use the prefix ".well-known" for their paths, making the combined size of the URI path options easily longer than the rest of the CoAP message.

This document establishes a CoAP option that allows abbreviating the path component of the request URI by encoding the path component as a single number. A registry is established to maintain the mapping between numbers and URI paths.

1.1. Motivating example

The design criteria for EDHOC [RFC9528] described in Section 2.11 of [I-D.ietf-lake-reqs-04] give a frame fragmentation limit of 47 bytes for a CoAP message payload for 6TiSCH and 51 bytes for some parameters (and implementations) of LoRaWAN, and imply high performance penalties of a CoAP message not fitting in a single frame. An EDHOC message 1 on its own carries a minimum of 37 bytes. The 18 bytes of an encoded "/.well-known/edhoc" URI path push the CoAP message size over either limit, whereas an equivalent Uri-Path-Abbrev option lets the message stay well below these limits.

1.2. Update to RFC7252

The use of a critical CoAP option that is not understood by the server has a CoAP response error code (4.02 Bad Option) assigned, which generally enables clients to retry without using the critical option. This mechanism is useful for the abbreviation mechanism of this document.

That mechanism got conflated into the mechanism of _rejecting_ a request established in Section 4.2 of [RFC7252] for handling unprocessable non-confirmable messages, which makes detection of missing option support less reliable.

Appendix A of this document updates Section 5.4.1 of [RFC7252] to repair the behavior of servers when they receive an unsupported critical option in a non-confirmable message.

1.3. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document assumes basic familiarity with CoAP ([RFC7252]), in particular its Uri-* options.

The term "expand" is used to refer to the process of translating a Uri-Path-Abbrev option into an equivalent sequence of Uri-Path options. The term "abbreviate" is used to refer to the reverse process of translating a sequence of one or more Uri-Path options into a single Uri-Path-Abbrev option.

2. The Uri-Path-Abbrev option

The Uri-Path-Abbrev option (short for "URI path, abbreviated") expresses a request’s URI path in a more compact form.

The Uri-Path-Abbrev value represents a particular URI path, and is thus equivalent to any number of Uri-Path options. Those paths are typically in a "/.well-known" location as described in [RFC8615]. The numeric option values are coordinated by IANA in the Uri-Path-Abbrev registry established in this document in Section 6.2.

No.	C	U	N	R	Name	Format	Len.	Default
CPA13	x				Uri-Path-Abbrev	uint	0-4	(none)

Table 1: Uri-Path-Abbrev Option Summary (C = Critical, U = Unsafe, N = NoCacheKey, R = Repeatable)

```
// 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.  There is one occurrence of the value 13 encoded
// in a1270d, which, if the number were to change, would need
// updating.
```

The option is critical, safe-to-forward, part of the cache key, non-repeatable and used in CoAP requests. Table 1 summarizes these, extending Table 4 of [RFC7252]). Its OSCORE treatment is as Class E ([RFC8613]).

The option has an integer value from the registry established in Section 6.2.

Apart from the format and repeatability, the option’s properties only deviate from the Uri-Path (for which it stands in) in that this option is safe to forward. This has consequences for its interactions with the Proxy-Uri option, but these consequences are generally desirable: It allows the option to be used with CoAP proxies that do not implement the option.

2.1. Server processing

A CoAP server receiving this option processes it like the equivalent sequence of Uri-Path options.

A server that supports a specific Uri-Path-Abbrev value MUST also support the equivalent request where the URI path is composed of Uri-Path options.

A server receiving the option with an unknown value MUST treat it as an unprocessable critical option, returning a 4.02 Bad Option response, and MUST NOT return a 4.04 Not Found response, because the equivalent path may be present on the server.

Since CoAP clients that use the option are usually aware of the possibility of failure, there is no need to provide a diagnostic payload in the error (as is generally recommended in Section 5.4.1 of [RFC7252]). A machine-readable (and, albeit beyond the scope of this document, actionable) response is described in Section 3.1.1 of [RFC9290]: the server can set Content-Format 257 in the response and send the payload a1270d, which is the CBOR encoding for the CoAP problem detail "Unprocessed CoAP option" with the value CPA13.

2.2. Client processing

A CoAP client can use the option instead of one or more Uri-Path option(s) if there is a suitable Uri-Path-Abbrev value that can express the requested URI path.

The option SHOULD only be sent when it is known that the CoAP server has support for the concrete value. This knowledge is typically not learned/discovered, but follows from other specifications mandating support for it.

A client can also send a value if it is merely likely that the server supports it. (The decision threshold varies by application, but generally it’s closer to 95% than a mere more-likely-than-not). This is called "tentative use" of the option.

In that case, the client needs to reliably detect failure of the option processing, and needs to fall back to repeating the request with the Uri-Path spelled out (using Uri-Path options), to operate reliably.

The client can expect that a server which does not support the Uri-Path-Abbrev option responds with 4.02 Bad Option. Diverging behavior has been allowed until the changes in Appendix A have been made. To account for older servers, the full set of reactions to expect is:

1. A 4.02 Bad Option response.

2. A 5.02 Bad Gateway response caused by a proxy that received a RST message or lack of response.

3. Having sent a Non-confirmable request message: A RST message.

4. Having sent a Non-confirmable request message: Not receiving a response.

Some of the complexity of detecting lack of server-side support (items 3 and 4) can be avoided by not using the option with Non-confirmable requests in tentative use. Clients that know that the server supports _any_ Uri-Path-Abbrev value can trust the server to reliably produce 4.02 Bad Option for other Uri-Path-Abbrev values.

As CoAP multicast requests generally do not result in errors being returned, tentative use is not available for multicast requests.

A generic client implementation SHOULD NOT use this option without explicit instructions from a higher layer or the known specification of the numeric value, because conditions for tentative use are generally not met in this case.

2.3. Proxy processing

A proxy MAY expand into Uri-Path options, or abbreviate to a Uri-Path-Abbrev option, before consulting its cache.

It MAY expand a Uri-Path-Abbrev option before forwarding, in particular if it has reason to assume that the option is not understood by the receiver. Like a generic client, it SHOULD NOT abbreviate without good reason; and if it does, it MUST be able to fall back to the expanded form upon failure, as to not introduce unexpected errors to the client.

A proxy that knows the Uri-Path-Abbrev option but not the concrete value at hand SHOULD forward a request unmodified, which is the behavior it would apply if it did not know the option. A valid exception where the proxy can reject the request instead is when the proxy is tasked with enforcing access control (see Section 5).

When cross-proxying to protocols that cannot transport this option (such as HTTP), the proxy needs to expand the path always.

2.4. Interaction with other options

The option is mutually exclusive with the Uri-Path option. Receiving both options in a single request, a server MUST treat the Uri-Path-Abbrev option as a critical request option that could not be processed.

The Uri-Path-Abbrev option MUST NOT be used in combination with the Proxy-Uri option (or the similar Proxy-CRI option (of [I-D.ietf-core-href])) by clients.

When a request gets forwarded through multiple proxies, any of them might convert the Uri-* options (but, when unaware of its significance, not Uri-Path-Abbrev) into a single Proxy-Uri/-CRI option. This Proxy-Uri/-CRI conversion will get reverted to Uri-* options before or at the final hop where the final hop is not proxy forwarding. It is thus generally inconsequential to client and server that Proxy-Uri/-CRI and Uri-Path-Abbrev occur in the same message in such a case.

Endpoints that process both the Proxy-Uri/-CRI and the Uri-Path-Abbrev option (which is, servers that are not forwarding like proxies, but are regarded as proxies by other proxies), MUST logically decompose the Proxy-* options before processing the Uri-Path-Abbrev option, which entails an error response if both a path segment in the Proxy-* option and Uri-Path-Abbrev are present.

2.5. Choice of the option number

// RFC-Editor: Please remove this section during publication. It is
// valuable only to the working group and designated experts who
// manage the limited resource of option numbers, and stays available
// for future documents that may want to apply similar rationale
// throughout the draft versions.

The option’s desired properties (critical, safe-to-forward, part of the cache key) limit the available number space to patterns ending with 0bXXX01 where XXX is not 111. All options encodable with a short (1+0-byte) delta, i.e. <= 12, are already assigned. Usually, we’d then look at other options this is typically combined with, and if there are none (as is the case here), go for a large value in the next more efficient (1+1-byte) space so that the small-but-quite-not-1+0-byte numbers stay usable as 1+0-byte options when combined in their "favorite pairings". (This was done with the EDHOC option [RFC9528] which is usually paired with the OSCORE option [RFC8613]).

However, in this case, there is an extra concern: The option number should also be smaller than Uri-Query, so that processing the options in a linear fashion can happen as it would happen with Uri-Path: the path gets processed first, setting up a state machine to process the query. As Uri-Query has option number 15, the value 13 is the only good choice for the Uri-Path-Abbrev option.

3. Initial Uri-Path-Abbrev values

This document registers values for the following well-known URIs:

* /.well-known/core

* /.well-known/rd (see [RFC9175])

* /.well-known/edhoc (see [RFC9528])

* For EST ([RFC9148]):

  - /.well-known/est/crts

- /.well-known/est/sen
- /.well-known/est/sren
- /.well-known/est/skg
- /.well-known/est/skc
- /.well-known/est/att

EST does allow using other paths, such as different root resources or arbitrary labels; for those, no abbreviations are supported in this document.

* For [I-D.ietf-anima-constrained-voucher]:

- /.well-known/brski/es
- /.well-known/brski/rv
- /.well-known/brski/vs

Note that the core and rd paths are commonly used together with Uri-Query options.

4. Implementation Status

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".

* aiocoap https://christian.amsuess.com/tools/aiocoap/ (https://christian.amsuess.com/tools/aiocoap/)

  A general-purpose implementation of CoAP for unconstrained sytems, published under MIT License.

  In its current main branch, the implementation covers the server side of this specification, applying expansion automatically before looking up which resource to serve. For client, all it provides is the option field where to place a number if the application decides it is suitable, relying on the client application to perform the fallback.

  It implements version ietf-core-uri-path-abbrev-01. Implementation experience: generally straightforward.

  The biggest trouble during implementation was that originally, it was attempted to give the server application access to information on whether or not Uri-Path-Abbrev was used. This was resolved by just not exposing the information -- after all, that is probably good layering practice anyway.

  Contact information: Christian Ams端ss (author), updated 2025-09-26

* libcoap (preliminary)

  A report on a yet unpublished implementation and related tests of ietf-core-uri-path-abbrev-01 in libcoap has been submitted at https://github.com/core-wg/uri-path-abbrev/issues/25 (https://github.com/core-wg/uri-path-abbrev/issues/25) on 2025-10-01.

5. Security Considerations

Having alternative expressions for information that is input to policy decisions can be problematic when the mechanism performing the check has a different interpretation of the presented data than the mechanism at time of use. That concern is not new to this document: Both the Proxy-Uri of [RFC7252] and the Proxy-Cri option of [I-D.ietf-core-href] have the same properties in that regard. The appropriate behavior is for policy checkers to reject any request that contains critical options that are not understood; the application protected by the checker may provide the checker with an allow-list of options that it will treat as unchecked input.

6. IANA Considerations

6.1. CoAP option: Uri-Path-Abbrev

IANA is requested to enter one option into the CoAP Option Numbers registry in the CoRE Parameters registry group:

* Number: CPA13
* Name: Uri-Path-Abbrev
* Reference: this document

6.2. Uri-Path-Abbrev registry

IANA is requested to establish a new registry in the CoRE parameters registry group. The value of the first Uri-Path-Abbrev option in a CoAP request corresponds to a URI path entry in this registry.

The policy for adding any value is IETF Review (as described in [RFC8126]). Change control for the registry follows this document's publication stream. Initial values for the registry are given in Table 2.

The required fields for each registration are:

* Option value.

  A non-negative integer (uint) that can be expressed in 32 bits, unique within this registry.

  All positive values whose most significant bit of the most significant non-zero byte is '1' are reserved.

  The Python invocation python3 -c 'print("reserved" if (250).bit_length() % 8 == 0 else "unreserved")' can be used to quickly test this property for any positive number (250 in this example).

* Expanded path.

  This text is the URI path (starting with /) that the option is expanded to.

* Reference.

  A document that requested the allocation.

Reviewer instructions:

The reviewer is instructed to be frugal with the 128 values that correspond to a single-byte option value, focusing on applications that are expected to be useful in different constrained ecosystems.

The expanded path is expected to be a well-known path ([RFC8615]), but it is up to the reviewers to exceptionally also admit paths that are not well-known.

Option value	Expanded path	Reference
0	/.well-known/core	Section 3 of this document
1	/.well-known/rd	Section 3 of this document, and [RFC9176]
2	/.well-known/edhoc	Section 3 of this document, and [RFC9528]
301	/.well-known/est/crts	Section 3 of this document, and [RFC9148]
302	/.well-known/est/sen	Section 3 of this document, and [RFC9148]
303	/.well-known/est/sren	Section 3 of this document, and [RFC9148]
304	/.well-known/est/skg	Section 3 of this document, and [RFC9148]
305	/.well-known/est/skc	Section 3 of this document, and [RFC9148]
306	/.well-known/est/att	Section 3 of this document, and [RFC9148]
401	/.well-known/brski/es	Section 3 of this document, and [I-D.ietf-anima-constrained-voucher]
402	/.well-known/brski/rv	Section 3 of this document, and [I-D.ietf-anima-constrained-voucher]
403	/.well-known/brski/vs	Section 3 of this document, and [I-D.ietf-anima-constrained-voucher]

Table 2: Initial values for the Uri-Path-Abbrev registry

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

7.2. 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-draft-numbers] Bormann, C., "Managing CBOR codepoints in Internet-Drafts", Work in Progress, Internet-Draft, draft-bormann-cbor-draft-numbers-07, 12 January 2026, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-draft-numbers-07>.

[I-D.ietf-anima-constrained-voucher] Richardson, M., Van der Stok, P., Kampanakis, P., and E. Dijk, "Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-30, 27 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-constrained-voucher-30>.

[I-D.ietf-core-corr-clar-03] Bormann, C., "Constrained Application Protocol (CoAP): Corrections and Clarifications", Work in Progress, Internet-Draft, draft-ietf-core-corr-clar-03, 22 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-corr-clar-03>.

[I-D.ietf-core-href]
Bormann, C. and H. Birkholz, "Constrained Resource Identifiers", Work in Progress, Internet-Draft, draft-ietf-core-href-30, 21 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-href-30>.

[I-D.ietf-lake-reqs-04]
Vuini, M., Selander, G., Mattsson, J. P., and D. Garcia-Carillo, "Requirements for a Lightweight AKE for OSCORE", Work in Progress, Internet-Draft, draft-ietf-lake-reqs-04, 8 June 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-reqs-04>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/rfc/rfc6690>.

[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>.

[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>.

[RFC8126] 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/rfc/rfc8126>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/rfc/rfc8132>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/rfc/rfc8613>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/rfc/rfc9148>.

[RFC9175] Amsss, C., Preu Mattsson, J., and G. Selander, "Constrained Application Protocol (CoAP): Echo, Request-Tag, and Token Processing", RFC 9175, DOI 10.17487/RFC9175, February 2022, <https://www.rfc-editor.org/rfc/rfc9175>.

[RFC9176] Amsss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/rfc/rfc9176>.

[RFC9290] Fossati, T. and C. Bormann, "Concise Problem Details for Constrained Application Protocol (CoAP) APIs", RFC 9290, DOI 10.17487/RFC9290, October 2022, <https://www.rfc-editor.org/rfc/rfc9290>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

Appendix A. RFC7252-5.4.1: Critical Options and Error Messages

This appendix is normative.

Section 4.2 of [RFC7252] introduces the concept of _rejecting_ a message, by sending back a RST (Reset) message or by silently ignoring the rejected message. This can deal with messages for which a node does not have (e.g., no longer has) the necessary context to process it, such as a response to a message that a node sent immediately before it rebooted.

The concept of rejecting a message is a quite powerful way to limit the complexity of dealing with a variety of error conditions. However, it seems Section 5.4.1 of [RFC7252] overuses this instrument for the case of non-confirmable messages.

Section 5.4.1 of [RFC7252] describes Critical Options, which, if not implemented/understood by a node, may require the return of server errors. For Confirmable messages containing requests, unrecognized critical options in requests are handled by a 4.02 (Bad Option) response, while those in Confirmable messages with responses and Acknowledgements are rejected.

> * Unrecognized options of class "critical" that occur in a Confirmable request MUST cause the return of a 4.02 (Bad Option) response. This response SHOULD include a diagnostic payload describing the unrecognized option(s) (see Section 5.5.2).
>
> * Unrecognized options of class "critical" that occur in a Confirmable response, or piggybacked in an Acknowledgement, MUST cause the response to be rejected (Section 4.2).

The 4.02 error response enables clients to perform discovery of whether a critical option is recognized by a server, but surprisingly only for Confirmable messages.

For unknown reasons, Section 5.4.1 of [RFC7252] then goes on to handle all Non-confirmable messages with unrecognized critical options by rejecting them:

> * Unrecognized options of class "critical" that occur in a Non-confirmable message MUST cause the message to be rejected (Section 4.3).

This deprives the sender of a Non-confirmable request of the information what caused the rejection. However, using Non-confirmable messages does not automatically mean that the recipient does not have the context needed to process the message.

This unexplained inconsistency has been present in [RFC7252] since its initial publication, apparently without causing much trouble. Section Section 2.2 of this document describes a situation where discovery of option support is more central to at least one use case; not being able to properly perform this discovery for Non-confirmable messages now emerges as an actual defect.

This defect can be remedied by applying this change:

INCORRECT:

> * Unrecognized options of class "critical" that occur in a Non- confirmable message MUST cause the message to be rejected (Section 4.3).

CORRECTED:

> | * Unrecognized options of class "critical" that occur in a
> | Non-confirmable request SHOULD cause the return of a 4.02
> | (Bad Option) response. This response SHOULD include a
> | diagnostic payload describing the unrecognized option(s)
> | (see Section 5.5.2).
> |
> | * Unrecognized options of class "critical" that occur in a
> | Non-confirmable response, or piggybacked in an
> | Acknowledgement, MUST cause the response to be rejected
> | (Section 4.3).

(This replacement text could be integrated in a less redundant way; the present proposal primarily aims at clarity.)

Of course, existing implementations will not immediately change their behavior, so an implementation of a discovery process in this document will still need to deal with uncertainty for the foreseeable future, but the likelihood will reduce over time. Client implementations that are not updated and actually rely on not receiving an error response in this case will simply reject the message, causing little downside.

Note that the SHOULD about diagnostic payloads is repeated here; such a mandate usually needs to provide more information about when this interoperability requirement does not need to be met. [RFC9290] now in many cases provides a better way to respond than a diagnostic payload; for conciseness, this observation is not included in the normative replacement text proposed above.

[The following section to be removed by the RFC editor.]

This technical change was originally developed as Section 2.2 of [I-D.ietf-core-corr-clar-03], and moved into this document for publication, so that it can be used for simplifications in describing tentative use, where the text would otherwise have had to reaffirm the original behavior.

Appendix B. Further development

This appendix is for information only.

Several possible further directions are anticipated in this document, but not specified at this point in time; they are left for further documents that update and thus compatibly extend this document.

In any case, server implementations that treat unknown option values or repeated Uri-Path-Abbrev options like any other critical unprocessable option (i.e., by responding with 4.02 Bad Option) support the transition to such an extension.

* Some values of the option might admit repetition of the option. A document that updates this document and the Uri-Path-Abbrev registry will need to specify how later occurrences of the option extend the series of equivalent Uri-Path options from the first value, or defer some of that decision to that first value's entry.

* The mechanism of expanding one option into another option might be expressed using the terminology of SCHC.

  Such a generalization is not aimed for in this document; authors of any future document providing such a framework are encouraged to provide an equivalent but machine-readable explanation of the mechanism specified here.

* The registry for Uri-Path-Abbrev values is set up such that first values can not have the most significant bit of the first byte set.

  This allows future documents to reuse the option for CBOR data items, e.g. the path component of a CRI [I-D.ietf-core-href]. Note that those CBOR data items can only use the major types 4 to 7 for the top-level item, but that includes all containers (arrays, maps and tags).

  Senders and recipients of this option do not need to concern themselves with that extension mechanism unless they implement it: As the first value is compared to known registry entries, any CBOR item contained in it will simply not match any known value. Should the working group decide not to use that extension point, the registry's policy can be relaxed to also allow values with that leading bit set.

* A future document may update this document to allow registering values that are allowed to use together with Uri-Path values (but at the time of writing, no examples are known by which such a design could be properly vetted). In particular, that update weakens the "MUST" in Section 2.4.

* This option is designed to stand in for the Uri-Path option alone, not for any other option; this simplifies its interaction rules.

In particular, application authors who seek to express Uri-Query options in a more concise or easier to process way are advised to avail themselves of the FETCH method introduced in [RFC8132].

Appendix C. Change log

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

Since ietf-core-uri-path-abbrev-03:

* Applied simplifications in handling of Proxy-Uri in combination of -03 more consistently:
  - Removed mandate that proxies that recognize Uri-Path-Abbrev perform conversion to Proxy-Uri expand it if known (they may still do that per more generic rules, but given that a later proxy would undo the Proxy-Uri packing, there’s no strong reason to, and it might hinder performance at a constrained server).
  - Limited requirements of processing a Proxy-Uri together with Uri-Path-Abbrev to those endpoints that are actually in a position where they need to do it: those that process both options. That requirement was previously a SHOULD (made sense in -02 when it’d have been done by any server just to cater for the odd proxy) and became a MUST (now that it’s only actionable for implementations that actually it).
* Editorial fixes

Since ietf-core-uri-path-abbrev-02:

* Added normative appendix fixing an oversight in RFC7252 (moving in from corr-clar). This allows limiting the handling of exotic cases in tentative use.
* Named Uri-Path-Abbrev as the unprocessed option when it occurs together with Uri-Path.
* Processing a mixed Proxy-Uri / Uri-Path-Abbrev is now an error if a path component is present in the former. Text was changed to indicate more strongly that seeing a Proxy-Uri and Uri-Path-Abbrev together is an exotic case even when there is no conflict.
* Formal definitions of abbreviate/expand.
* Editorial enhancements.

Since ietf-core-uri-path-abbrev-01: Processing WGA review input and cleanup.

* Using Uri-Path-Abbrev without knowing it will be supported now has a term ("tentative use") and was reworded.

* The "SHOULD" to support fallback mode was lifted to "SHOULD only be [used when the server is known to support it]", with the recovery being a factual necessity for reliability in tentative use.

* The fallback detection was extended to account for possible reactions to NONs (namely, RST, not replying at all, and 5.02).

* Use with multicast was limited to non-tentative use.

* Added reference to libcoap work.

* Added pointer to RFC9290 (Problem Details) error messages, discouraging diagnostic payloads.

* Sections on future work were unified in the appendix.

* Appendix on open questions was moved into the issue tracker at https://github.com/core-wg/uri-path-abbrev/issues/ (https://github.com/core-wg/uri-path-abbrev/issues/).

* Cleanup of IANA considerations where -01 previous changes were not accounted for.

* "Choice of option number" spelled out and set up for removal before publication.

* Editorial changes.

Since ietf-core-uri-path-abbrev-00: Processing previous two interims.

* Rename option to Uri-Path-Abbrev.

* Allocate per-resource codes for EST and cBRSKI.

* Allocate code for EDHOC.

* Defer repeated use to future extensions.

* Rearrange content to have dedicated server, client and proxy subsections for option processing.

* Establish that generic clients SHOULD NOT use this without reason.

* More explicit language for proxies, including cross-proxies.

* Add introduction and motivating example.

* Add RFC7942 Implementation Status section.

Since draft-amsuess-core-shopinc-02:

Adopted into WG unmodified as I-D.ietf-core-uri-path-abbrev

Since draft-amsuess-core-shopinc-01: Processing 2025-08-27 interim.

* Document is standards track.

* Change name of the option from Short-Uri-Path to Uri-Path-Abbr.

* Close question of whether use of option 13 is justified (it is).

* Minor editorials.

Since draft-amsuess-core-shopinc-00:

* Switched option type from opaque to uint (retaining the lockout for values that look like CBOR arrays/maps).

* MCR joined as author.

* Added initial values for BRSKI and EST.

* Allow 4.04 responses.

* Add guidance for choosing prefixes and rules.

* Large editorial changes.

Acknowledgments

Discussion with Eski Dijk led to the creation of the document, he questioned choices until the design was simple enough, and also provided editorial input. Carsten Bormann provided Appendix A, as well as useful input on shaping the registry. Jon Shallow provided much input, in particular around gaps in the fallback process.

Contributors

Carsten Bormann
Universitt Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

Authors' Addresses

Christian Amsss
Austria
Email: christian@amsuess.com

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca