Network Working Group
Internet-Draft
Updates: 6690, 7252, 7641, 7959, 8132, 8323 (if approved)
Intended status: Standards Track
Expires: 20 September 2026

C. Bormann
Universit辰t Bremen TZI

19 March 2026

Constrained Application Protocol (CoAP): Corrections and Clarifications
draft-ietf-core-corr-clar-04

Abstract

RFC 7252 defines the Constrained Application Protocol (CoAP), along with a number of additional specifications, including RFC 7641, RFC 7959, RFC 8132, and RFC 8323. RFC 6690 defines the link format that is used in CoAP self-description documents.

Some parts of the specification may be unclear or even contain errors that may lead to misinterpretations that may impair interoperability between different implementations. The present document provides corrections, additions, and clarifications to the RFCs cited; this document thus updates these RFCs. In addition, other clarifications related to the use of CoAP in other specifications, including RFC 7390 and RFC 8075, are also provided.

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-corr-clar/.

Discussion of this document takes place on the core 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/corrclar.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction

[RFC7252] defines the Constrained Application Protocol (CoAP), along with a number of additional specifications, including [RFC7641], [RFC7959], [RFC8132], and [RFC8323]. [RFC6690] defines the link format that is used in CoAP self-description documents.

During implementation and interoperability testing of these RFCs, and in their practical use, some ambiguities and common misinterpretations have been identified, as well as a few errors.

The present document summarizes identified issues and provides corrections needed for implementations of CoAP to interoperate, i.e., it constitutes an update to the RFCs referenced. This document also provides other clarifications related to common misinterpretations of the specification. References to CoAP should, therefore, also include this document.

In addition, some clarifications and corrections are also provided for documents that are related to CoAP, including RFC 7390 and RFC 8075.

1.1. Process

1.1.1. Original text

The present document is an Internet-Draft, which is not intended to be published as an RFC quickly. Instead, it will be maintained as a running document of the CoRE WG, probably for a number of years, until the need for new entries tails off and the document can finally be published as an RFC. (This paragraph to be rephrased when that happens.)

The status of this document as a running document of the WG implies a consensus process that is applied in making updates to it. The rest of this subsection provides more details about this consensus process.

(Consensus process TBD, but it will likely be based on an editor's version in a publicly accessible git repository, as well as periodic calls for consensus that lead to a new published Internet-Draft.)

1.1.2. Proposed text based on IETF 117 and 2023-08-30 CoRE WG interim discussion

This section describes the process that will be used for developing the present document (called "-corr-clar" colloquially).

This process might be revised as its execution progresses.

0. (Done as of this a draft): include the present process proposal. The document can then already be considered for WG adoption.

1. Go through the following available material and revise/create Github issues at ISSUES (https://github.com/core-wg/corrclar/issues) as needed:

   * Existing issues at ISSUES (https://github.com/core-wg/corrclar/issues)

     - More to be opened by Jon Shallow regarding Block-wise, see JON-ISSUES (https://datatracker.ietf.org/doc/minutes-interim-2023-core-11-202307051400/#attacks-on-the-constrained-application-protocol-coap-christian-amsuss-jon-shallow)

   * CoAP FAQ at the CoRE WIKI WIKI-FAQ (https://github.com/core-wg/wiki/wiki/CoAP-FAQ)

     - Each point likely to become a new, short issue

2. Categorize the Github issues at ISSUES (https://github.com/core-wg/corrclar/issues) as to the topics they relate to, by tagging them.
   Completing a first round of this will be a task for a dedicated team.

3. For each issue or set of issues at ISSUES (https://github.com/core-wg/corrclar/issues), confirm with the CoRE WG and gather feedback from affected protocol designers/implementors if the issue is best to be:

   * Included and covered in -corr-clar, as is or revised

   * Simply omitted in -corr-clar

* Omitted in -corr-clar and left for a possible -bis document. (For example, this might be the case for some specific points related to RFC 7959.)

4. Reshape the -corr-clar document in order to reflect a sequence of pairs (Diagnosis, Therapy), where:

   * Diagnosis is the gist of a set of Github issues to cover, and

   * Therapy is the correction or clarification to address those.

   Even though at a high-level, the scope should be already clear by looking at the table of contents. That is, at this stage, there is no need to necessarily elaborate the Therapy in detail, but it is necessary to make a reader understand "what we are dealing with and taking which direction".

5. WG document work can then focus on improving the therapy parts, until all points are satisfactorily addressed and documented.

1.2. Terminology

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.

When a section of this document makes formal corrections, additions or invalidations to text in a referenced RFC, this is clearly summarized. The text from the RFC that is being addressed is given and labeled "INCOMPLETE", "INCORRECT", or "INCORRECT AND INVALIDATED", followed by the correct text labeled "CORRECTED", where applicable. When text is added that does not simply correct text in previous specifications, it is given with the label "FORMAL ADDITION".

Where a resolution has not yet been agreed, the resolution is marked PENDING.

In this document, a reference to a section in RFC nnnn is written as RFC nnnn-<number>, where <number> is the section number.

2. RFC 7252

2.1. RFC7252-1.2: Terminology (Request-URI)

[RFC7252] uses the term "request URI" repeatedly, but only provides a vague definition in Section 5.7.2 of [RFC7252]. Text should be added to the definitions in Section 1.2 (Terminology) of [RFC7252].

INCOMPLETE; FORMAL ADDITION (Section 1.2):

> Request URI: The URI that identifies a resource on a server intended to be addressed by a request; constructed from the context of the request combined with Options present in the request using the process defined in Section 6.5 (Composing URIs from Options) of [RFC7252], see Section 5.7.2 (Forward-Proxies) of [RFC7252] for further details. Related to the HTTP concept of "target URI"; see Section 7.1 (Determining the Target Resource) of [RFC9110]; previously called "effective request URI" in Section 5.5 (Effective Request URI) of [RFC7230].

PENDING.

Note that a similar, but distinct concept is the "base URI", relative to which relative URIs are resolved. This is more complex in CoAP than in HTTP because CoAP can multicast requests (Section 8 of [RFC7252]), expecting unicast responses; these need to be interpreted relative to the unicast source address from which the responses come.

Section 8.2 of [RFC7252] has:

> For the purposes of interpreting the Location-* options and any links embedded in the representation, the request URI (i.e., the base URI relative to which the response is interpreted) is formed by replacing the multicast address in the Host component of the original request URI by the literal IP address of the endpoint actually responding.

Similarly, Section 8.2.1 of [RFC7252] (Caching) says:

> A response received in reply to a GET request to a multicast group MAY be used to satisfy a subsequent request on the related unicast request URI. The unicast request URI is obtained by replacing the authority part of the request URI with the transport-layer source address of the response message.

Further discussion of a more generalized response concept can be found in [I-D.bormann-core-responses].

2.2. RFC7252-5.4.1: Critical Options and Error Messages

Appendix A of [I-D.ietf-core-uri-path-abbrev] points out that [RFC7252] was overly eager in _rejecting_ Non-confirmable request messages where a proper response might have carried useful problem detail information [RFC9290]. [I-D.ietf-core-uri-path-abbrev] contains an update to [RFC7252] that for this case instead specifies the use of 4.02 "Bad Option" responses.

As this is a technical change, it needs to be included in a standards-track RFC to become effective. The present document therefore does not repeat the information but simply points to [I-D.ietf-core-uri-path-abbrev], which both includes the update and also directly makes use of the updated functionality if it is available.

2.3. RFC7252-5.8: Piggybacking 4.05 Responses

Section 5.8 of [RFC7252] says:

INCORRECT:

> A request with an unrecognized or unsupported Method Code MUST generate a 4.05 (Method Not Allowed) piggybacked response.

However, a piggybacked response is carried in an ACK, which is only used if the request is a Confirmable message. If the request is a Non-confirmable message, the response can only be sent as a Separate Response (Section 5.2.2 of [RFC7252]).

CORRECTED:

> A request with an unrecognized or unsupported Method Code MUST generate a 4.05 (Method Not Allowed) response. This MUST be sent as a piggybacked response if the request is a Confirmable message (responses for Non-confirmable messages can only be sent as a Separate Response).
>
> Note that the response MAY be suppressed before actually being sent, e.g., when:
>
> * The server is able to determine that the request was sent to multiple recipients (e.g., over IP multicast) and the application does not deem that the response ought to be sent.
>
> * The request carried the CoAP No-Response Option [RFC7967] with an indication that 4.xx error responses are to be suppressed, and the server ultimately determines such a suppression to be appropriate and useful.

> |
> | While this specification is explicit in requiring a 4.05
> | (Method Not Allowed) response, the response code 5.01 (Not
> | Implemented) is also defined (Section 5.9.3.2 of [RFC7252]) and
> | might be sent as a catch-all by a server with limited precision
> | in error handling. Although this response code is not as clear
> | in designating the problem a client error, the client SHOULD be
> | prepared to handle such a response, but possibly in a degraded
> | way.

PENDING.

// The text so far doesn't discuss proxies that forward 4.05
// responses; as an ACK may already have been sent by the proxy, this
// may convert a piggy-backed response by the origin server to a
// separate one.

2.4. RFC7252-5.10.1/6.1: Query Parameters

Section 3.4 of [RFC3986] explains the query component of a URI as follows:

> | The query component contains non-hierarchical data that, along
> | with data in the path component (Section 3.3), serves to identify
> | a resource within the scope of the URI's scheme and naming
> | authority (if any). [...]

So there is no technical difference between a path and a query component in a URI except that the path is hierarchical, and the query is non-hierarchical. Both combine with scheme and authority to identify the resource, and changing any of these leads to a different resource.

[RFC7252] generally follows this definition, but has a few passages where it is somewhat questionable whether they fully agree:

Section 5.10.1 (Uri-Host, Uri-Port, Uri-Path, and Uri-Query) of [RFC7252] says:

> | [...] each Uri-Query Option specifies one argument parameterizing
> | the resource.

(Analog text about Location-Query is in Section 5.10.7 (Location-Path and Location-Query) of [RFC7252].)

Similarly, Section 6.1 (coap URI Scheme) of [RFC7252] says:

> | The query serves to further parameterize the resource. [...]

This could be read to say that the _same_ resource can be accessed with different parameters in the Uri-Query options. It is likely that these passages were meant in the sense of "parameterize the resource name", i.e., changing the parameters does indeed lead to a different resource.

So this could be clarified by applying this change in these three places:

INCORRECT: further parameterize the resource

CORRECTED: further parameterize the resource name

However, the view that the query component supplies parameters to a request on a resource that exists independent of these parameters is widely held by implementers in the "big web" (HTTP) world, even if [RFC9110] strictly follows [RFC3986].

This view seems to be fueled by the way that an application may use (Section 17.9 (Disclosure of Sensitive Information in URIs) of [RFC9110])...

> | client-side mechanisms to construct a target URI out of user-
> | provided information, such as the query fields of a form using GET

This view also seems to be suggested to some by the way query parameters are used in specifications such as [I-D.ietf-core-conditional-attributes]; implementations of CoAP servers also often provide primitives to set up a single "resource handler" that bundles requests to a specific path and receives the query parameters as additional request parameters [COAP-RESOURCE].

[BCP190] establishes guidelines with respect to "URI Design and Ownership". Section 2.4 of RFC 8820 [BCP190] specifically discusses the query component of the URI. On one hand, it says:

> | Applications can specify the syntax of queries for the resources
> | under their control.

On the other hand, it warns:

> | Extensions MUST NOT constrain the format or semantics of queries, to avoid collisions and erroneous client assumptions. For example, an Extension that indicates that all query parameters with the name "sig" indicate a cryptographic signature would collide with potentially preexisting query parameters on sites and lead clients to assume that any matching query parameter is a signature.

It also acknowledges:

> | Per the "Form submission" section of [HTML5], HTML constrains the syntax of query strings used in form submission. New form languages are encouraged to allow creation of a broader variety of URIs (e.g., by allowing the form to create new path components, and so forth).

[I-D.ietf-core-conditional-attributes] is a specification that defines a set of query parameters. It can be adopted by an "Application" in the sense of [BCP190]. The current discussion goes in the direction of providing an interface type (for use in [RFC6690] if=) that a server can declare in resource discovery to indicate to a client that the conditional query parameters interface is available.

Section 3.3 of [I-D.irtf-t2trg-rest-iot] contains a brief discussion of the role of query parameters in URIs. It points to Section 3.3.4 of [RFC6943], which discusses identifier comparison and points out that

> | [...] it is unspecified whether the order of values matters.

Section 3.3 of [I-D.irtf-t2trg-rest-iot] further mentions that in some implementations,

> | they might even be re-ordered, for instance by intermediaries.

[I-D.ietf-core-conditional-attributes] is designed to allow evolution of the set of conditional query parameters and provides a registry for the registration of additional ones. To facilitate the introduction of new parameters, it has been discussed whether [I-D.ietf-core-conditional-attributes] should adopt an "ignore unknown" policy. Since that policy would not necessarily apply to query parameters outside the conditional set, it would require a test whether a parameter is a conditional query parameter. The names of all parameters initially registered start with "c.", which might be a convention that might be part of the if= interface type.

Any "ignore unknown" policy raises the question whether there, conversely, need to be "must understand" exceptions, here chosen by the client. For instance, the client might use the parameter name "C.pmin" instead of "c.pmin" to indicate that it wants the request to fail if its preference for a minimum period between notifications cannot be fulfilled.

In summary: In the definitions of URIs and of HTTP, the URI query parameters are not much different from path components. In practice, implementations tend to provide ways to handle query parameters more in terms of additional parameters to a single resource handler that handles a number of related URIs, differing in query parameters only. There appears to be little to be gained by discussing this more in the documentation (outside the present document); however, the duality between these two perspectives on query parameters needs to be kept in mind in discussions.

2.5. RFC7252-5.10.5: Max-Age

In the discussion of [RFC8516], a comment was made that it would be needed to define the point in time relative to which Max-Age is defined. A sender might reference it to the time it actually sends the message containing the option (and paragraph 3 of RFC7252-5.10.5 indeed requests that Max-Age be updated each time a message is retransmitted). The receiver of the message does not have reliable information about the time of sending, though. It may instead reference the Max-Age to the time of reception. This in effect extends the time of Max-Age by the latency of the packet. This extension was deemed acceptable for the purposes of [RFC8516], but may be suboptimal when Max-Age is about the lifetime of a response object.

INCOMPLETE:
The value is intended to be current at the time of transmission.

PENDING.

2.6. RFC7252-6.4: Decomposing URIs into Options

[Err4895] notes (text updated with newer link):

| The current specification for decomposing a URI into CoAP Options
| (Section 6.4) is correct; however the text may still be unclear to
| implementers who may think that the phrase "not including the
| delimiting slash characters" means simply omitting a trailing
| slash character in the URI path. This is incorrect. See the
| discussion outcome in email thread
| https://mailarchive.ietf.org/arch/msg/core/

> | vqOiUreodGXqWZGeGOTREChCsKY/
> | (https://mailarchive.ietf.org/arch/msg/core/
> | vqOiUreodGXqWZGeGOTREChCsKY/).

[Err4895] proceeds to propose adding another note at the end of Section 6.4 of [RFC7252]. A slightly updated version of the proposed note might be:

FORMAL ADDITION at the end of Section 6.4 of [RFC7252]:

> Also note that a trailing slash character in the <path> component represents a separate, zero-character path segment (see the ABNF in Section 3.3 of [RFC3986]). This is encoded using a Uri-Path Option of zero length. The exception at the start of step 8 means that no such zero-character path segment is added for a trailing slash that immediately follows the authority (host and port).
>
> > | This exception means that, for a CoAP server, no difference
> > | is visible between a request that was generated from the URI
> > | coap://authority/ and one that was generated from the URI
> > | coap://authority — in both cases, there is no Uri-Path
> > | Option in the request. (The URI continues to be parsed as
> > | defined: e.g., for the URIs coap://authority/? and
> > | coap://authority?, there is no Uri-Path Option but a single
> > | Uri-Query Option that carries an empty query component.)
> > |
> > | Note that the exception at the start of step 8 does not mean
> > | that a client cannot create a request with a single empty
> > | Uri-Path Option; such a request just cannot be generated
> > | from a URI because of the algorithm given here. It also
> > | does not mean that a server is compelled to treat a request
> > | with such a single empty Uri-Path Option in the same way as
> > | one without any Uri-Path Option — the exception at the start
> > | of step 8 is only about the process of generating a sequence
> > | of CoAP Options from a URI.
> > |
> > | The exception only applies to initial Uri-Path Options. So
> > | for coap://authority/foo, a single Uri-Path Option with the
> > | value foo is generated, while for coap://authority/foo/ that
> > | Uri-Path Option is followed by an empty Uri-Path Option (an
> > | established idiom for a collection resource).
> > |

> | Similarly, there is a difference between requests generated
> | from coap://authority/, coap://authority//, and
> | coap://authority///, respectively: The first has no Uri-Path
> | Options (due to the special exception); the second, two
> | (empty ones); the third, three (empty ones). No server is
> | compelled to offer resources under URIs with multiple empty
> | path name components, which would generally be considered
> | weird.

2.7. RFC7252-7.2.1: "ct" Attribute (content-format code)

As has been noted in [Err5078], there is no information in [RFC7252] about whether the "ct" target attribute can be present multiply or not.

The text does indicate that the value of the attribute MAY be "a space-separated sequence of Content-Format codes, indicating that multiple content-formats are available", but it does not repeat the prohibition of multiple instances that the analogously structured "rt" and "if" attributes in Sections 3.1 and 3.2 of [RFC6690] have.

This appears to be an oversight. Published examples in Section 4.1 of [RFC9148] and Section 4.3 of [RFC9176] further illustrate that the space-separated approach is generally accepted to be the one to be used. There is no gain to be had from allowing both variants, and it would be likely to cause interoperability problems.

At the 2022-11-23 CoRE WG interim meeting, there was agreement that [Err5078] should be marked "VERIFIED", which was done on 2023-01-18.

INCOMPLETE; FORMAL ADDITION:

> The Content-Format code attribute MUST NOT appear more than once in a link.

2.8. RFC7252-9.1.1/9.1.2: (match boxing)

Sections 9.1.1 and 9.1.2 of [RFC7252] provide details about using CoAP over DTLS connections; in particular they restrict both message-id matching and request/response matching to within a single combination of DTLS session/DTLS epoch.

At the time, this was a decision to be very conservative based on the wide variety of security implications a new DTLS epoch might have (which also were not widely understood, e.g., for a re-authentication). The normally short time between a request and a response made this rather strict boxing appear acceptable.

However, with the Observe functionality [RFC7641], it is quite likely that significant time elapses between a request and the arrival of a notification that is sent back as a response, causing a need for the latter to use a different box from the original request.

Also, additions to CoAP such as CoAP over connection-oriented transports [RFC8323] and OSCORE [RFC8613] create similar matching boxes that also do not fit certain likely use cases of these additions (e.g., with short-lived TCP connections as discussed in Section 4.3 of [RFC9006]).

The match boxing semantics of the current protocol are clearly defined, but can be unsatisfactory given the current requirements.

This calls for careful design choices and enhancements when developing extensions for CoAP or protocols and methods applicable to CoAP, such as in the cases overviewed in the following Section 2.8.2 and Section 2.8.3.

2.8.1. DTLS with Connection ID

PENDING:

> | Protocol mechanisms that have been defined for stitching
> | together connections or phases of an underlying connection,
> | such as Connection Identifiers for DTLS 1.2 [RFC9146], may
> | enable keeping the session/epoch unchanged and even to change
> | the transport address (ip-address/port), once appropriately
> | modified match boxing rules are specified for the stitching
> | mechanism. (These rules either need to be defined to be
> | implicitly active for any use of the mechanism or they may
> | require negotiation.)

2.8.2. OSCORE, KUDOS, and Group OSCORE

The security protocol Object Security for Constrained RESTful Environments (OSCORE) defined in [RFC8613] provides end-to-end security for CoAP messages at the application level, by using CBOR Object Signing and Encryption (COSE) [RFC9052]. In order to protect their communications, two peers need to have already established an OSCORE Security Context.

Appendix B.2 of [RFC8613] provides an example for a key update procedure, which two OSCORE peers can run for establishing a new shared OSCORE Security Context that replaces their old one. The recent key update protocol KUDOS [I-D.ietf-core-oscore-key-update] specifies how two OSCORE peers can establish a new shared OSCORE Security Context that replaces their old one, with a number of advantages compared to the method defined in Appendix B.2 of [RFC8613].

The security protocol Group Object Security for Constrained RESTful Environments (Group OSCORE) defined in [I-D.ietf-core-oscore-groupcomm] builds on OSCORE and protects group communication for CoAP [I-D.ietf-core-groupcomm-bis]. The management of the group keying material is entrusted to a Group Manager (see Section 3 of [I-D.ietf-core-oscore-groupcomm]), which provides the latest group keying material to a new group member upon its group joining, and can update the group keying material by performing a group rekeying.

The following discusses how OSCORE, KUDOS, and Group OSCORE position themselves with respect to the match boxing, the transport used underlying CoAP, and the renewal of the keying material.

2.8.2.1. Match Boxing

The security processing of (Group) OSCORE is agnostic of the value assumed by the CoAP Token and Message ID. Also, (Group) OSCORE can be seamlessly used in the presence of (cross-)proxies that will change the value of the CoAP Token and Message ID on the different communication legs. This does not affect the security processing at the (Group) OSCORE endpoints.

Before any security processing is performed, the only use that (Group) OSCORE makes of the Token value is on the CoAP client upon receiving a response, in order to retrieve the right Security Context to use for decrypting and verifying the response.

Even in case the Token value in a CoAP response is manipulated to induce a Request-Response matching at the client, there is no risk for the client to successfully decrypt the response instead of failing as expected. This is because, per Section 12.3 of [RFC8613], the OSCORE Master Secret of each OSCORE Security Context is required not only to be secret, but also to have a good amount of randomness.

Building on that, an HKDF is used to derive the actual encryption keys from the Master Secret and, optionally, from an additional Master Salt. Furthermore, for each OSCORE Security Context, the quartet (Master Secret, Master Salt, ID Context, Sender ID) must be unique. As per Section 3.3 of [RFC8613], this is a hard requirement and guarantees unique (key, nonce) pairs for the AEAD algorithm used.

In Group OSCORE, the Security Context extends that of OSCORE, and the same as above holds (see Sections 2, 2.2, and 13.11 of [I-D.ietf-core-oscore-groupcomm]).

Finally, (Group) OSCORE performs a separate secure match boxing under its own control, by cryptographically binding each protected request with all the corresponding protected responses. This is achieved by means of the COSE external_aad involved during the security processing of protected messages (see Section 5.4 of [RFC8613] and Section 4.4 of [I-D.ietf-core-oscore-groupcomm]).

2.8.2.2. Underlying Transport

The security protocol (Group) OSCORE does not have any requirement on binding the Security Context in use to specific addressing information used by the transport protocol underlying CoAP. What occurs below (Group) OSCORE with transport-specific addressing information is transparent to (Group) OSCORE, but it needs to work well enough to ensure that received data is delivered to (Group) OSCORE for security processing.

Consistent with the above, (Group) OSCORE does not interfere with any low-layer, transport specific information. Instead, it entrusts data to a Request-Response exchange mechanism that can rely on different means to enforce the Request-Response matching at the transport level (e.g., the 5-tuple, the CoAP Message ID, a file handle). Also, (Group) OSCORE does not alter the fact that a CoAP response needs to come from where the corresponding CoAP request was sent, which simply follows from using transports where that is a requirement.

Furthermore, two peers can seamlessly use (Group) OSCORE also in the presence of cross-proxies that change transport across different communication legs. This does not affect the security processing at the (Group) OSCORE endpoints.

Practically, (Group) OSCORE relies on the underlying CoAP implementation for obtaining received CoAP messages on which to perform the expected security processing.

Upon receiving a protected message, the recipient endpoint retrieves the OSCORE Security Context to use for decryption based on key identifier information specified in the CoAP OSCORE Option of protected requests, and on the value of the CoAP Token of protected responses.

In OSCORE, the key identifier information in request messages is typically limited to a "kid", with a value the OSCORE Sender ID associated with the message sender. In case Sender IDs are not unique among different OSCORE Security Contexts stored by the same peer, it is possible to disambiguate by additionally using a "kid context" identifying the OSCORE Security Context as a whole. Instead, response messages are not required to convey key identifier information, as the client can rely on the Token conveyed in the response for retrieving the Security Context to use (see above).

In Group OSCORE, the key identifier information in request messages always includes also a "kid context", whose value is used as identifier of the OSCORE group associated with the Security Context to use for security processing of the exchanged message. Response messages are also required to convey a "kid" as key identifier information (i.e., the OSCORE Sender ID associated with the message sender), if the corresponding request was protected with the group mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]) .

Some particular uses of (Group) OSCORE enable to build OSCORE-based tunneling [I-D.ietf-core-oscore-capable-proxies]. In such a case, a CoAP server might have to enforce that some OSCORE Security Contexts are not just looked up by a "kid" and similar key identifier information from the CoAP OSCORE Option in the incoming request to decrypt. Such a lookup should also rely on the alleged client’s address, or on an alternative identifier of the tunnel from which the request came from.

2.8.2.3. Key Update

Updating an OSCORE Security Context does not change or interfere with the values of the Token or Message ID used in the exchanged CoAP messages. However, if long-term exchanges are involved (e.g., CoAP Observations [RFC7641]), one has to be careful to ensure that updating the Security Context does not impair the security properties of (Group) OSCORE or result in other security vulnerabilities.

The following provides more details about key update, separately for OSCORE, KUDOS, and Group OSCORE.

* OSCORE: [RFC8613] tacitly assumes that two peers terminate any ongoing CoAP Observation that they still have ongoing, upon installing a new OSCORE Security Context, irrespective of the method used to perform the key update.

  On these premises, a belated response protected with the old OSCORE Security Context will fail decryption, as that Security Context is not available anymore on the receiving client.

* KUDOS: The key update protocol KUDOS allows the two OSCORE peers to negotiate about preserving their ongoing CoAP Observations across the performed key update. If and only if both peers agree to do that during an execution of KUDOS, their Observations will remain active after installing the new OSCORE Security Context, which the two peers will use from then on to protect their exchanged Observe notifications.

  Furthermore, irrespective of the method used to perform a key update, Section 3 of [I-D.ietf-core-oscore-key-update] updates the security protocol OSCORE [RFC8613] in order to prevent security issues that can arise from misbinding a request and a response, when those are protected with two different OSCORE Security Contexts.

  Such an update to the OSCORE protocol requires a server to include its own Sender Sequence Number as Partial IV of an outgoing response, when protecting it with a Security Context different from the one used to protect the corresponding request. An exception safely applies to the response messages that are sent when running the key update procedure defined in Appendix B.2 of [RFC8613].

* Group OSCORE: The Group Manager can distribute new group keying material to the members of an OSCORE group, by performing a group rekeying. When receiving updated group keying material from the Group Manager, either upon joining the group or by participating in a group rekeying, a group member uses that material to install a new, commonly shared Group OSCORE Security Context, which replaces the old one (if any is stored).

  Also, Group OSCORE makes it possible for group members to safely preserve their ongoing active requests (e.g., CoAP Observations), also across the establishment of new Group OSCORE Security Contexts. This is achieved by virtue of how the Group Manager assigns and maintains the identifiers of OSCORE groups (see Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm]).

Furthermore, analogous to the update that [I-D.ietf-core-oscore-key-update] makes on the OSCORE protocol with respect to protecting responses, Group OSCORE prevents security issues that can arise from misbinding a request and a response, when those are protected with two different Group OSCORE Security Contexts.

In the same way specified in Section 3 of [I-D.ietf-core-oscore-key-update] for OSCORE, Group OSCORE requires a server to include its own Sender Sequence Number as Partial IV of an outgoing response, when protecting it with a Security Context different from the one used to protect the corresponding request (see Sections 8.3 and 9.5 of [I-D.ietf-core-oscore-groupcomm]).

2.8.3. Eclipse/Californium

Enhancements may be called for optimizations such as Eclipse/Californium EndpointContextMatcher [CF-MATCHER] might not work properly unless both sides of the communication agree on the extent of the matching boxes. Mechanisms to indicate capabilities and choices selected may need to be defined; also, a way to define the progression of matching boxes needs to be defined that is compatible with the security properties of the underlying protocols. This may require new efforts, to be initiated based on some formative contributions.

PENDING.

Relevant starting points for retrieving existing discussion of this issue include:

* https://mailarchive.ietf.org/arch/msg/core/biDJ8n4w0kBQATzyh9xHlKnGy1o/ (https://mailarchive.ietf.org/arch/msg/core/biDJ8n4w0kBQATzyh9xHlKnGy1o/) ("DTLS and Epochs")

* https://mailarchive.ietf.org/arch/msg/core/TsyyKIHQ1FJtAvAo0ODNEt2Ileo/ (https://mailarchive.ietf.org/arch/msg/core/TsyyKIHQ1FJtAvAo0ODNEt2Ileo/) ("Connection ID")

* https://github.com/core-wg/corrclar/issues/9 (https://github.com/core-wg/corrclar/issues/9) ("Clarify/revise language around epochs in section 9.1.1 #9")

2.9. RFC 7252-9.1/11.3: Handling outdated addresses and security contexts

INCOMPLETE: Tools for mitigating these scenarios were unavailable when CoAP originally was specified, and are now explained.

PENDING.

Information obtained about an established communication partner can experience continuity interruptions and become obsolete. This can happen on different levels: For example, return routability information is lost when client's IP address changes; and information about whether a request was already handled can be lost when an OSCORE context is recovered as described in its Appendix B.1.2 of [RFC8613]. No matter the cause of the loss, a server still needs to maintain its security and amplification mitigation properties.

The concerns addressed in the following subsections are independent on a specification level, but are described together because they can be addressed with the same tools -- moreover, a single round trip of using Echo in a response can address both at the same time. In some scenarios, these are even expected to coincide.

(move) A safe option is always to reject the initial request and request confirmation. The RECOMMENDED way to do that is using CoAP's mechanism of sending a 4.01 (Unauthorized) response with an Echo option (where a subsequent request with the same Echo value proves to the server that the destination was reachable); clients SHOULD act to the Echo option as specified in [RFC9175]. Tools specific to a security protocol, such as RRC [I-D.ietf-tls-dtls-rrc] in case of DTLS, may be used. However, their use may depend on successful negotiation.

2.9.1. Amplification mitigation and return routability

CoAP servers have to mitigate the effects of traffic amplification as required in Section 11.3 of [RFC7252]; the maximum acceptable amplification factor of 3 is now the IETF consensus reflected for CoAP in Section 2.4 of [RFC9175], which can be summarized for this application as follows:

If the server has learned that the client is reachable on the request's sender address, it can send a response. Otherwise, if the response does not exceed the request's size by a factor of 3 (Section 2.4 of [RFC9175], item 3), the server can answer the request successfully, but should still include an Echo value, whose presence in the next request serves to confirm the client's address. Otherwise, a 4.01 (Unautorized) error response with an Echo value is sent.

Address verification is triggered both for new clients and for existing clients that changed their address. This situation can happen at any time, especially after a prolonged period of quiescence, regardless of the security protocol.

Verifying the client's address is not only crucial for mitigating amplification attacks [I-D.irtf-t2trg-amplification-attacks], but also to avoid traffic misdirection. Section 7 of [I-D.ietf-tls-dtls-rrc] describes options for how to use RRC messages to distinguish different cases. A 4.01 response with Echo can perform some of the same functions, with the Echo value replacing the RRC cookie. However, it does not offer a way to distinguish between non-preferred and preferred paths. Where that distinction matters, RRC provides the right tools to make it.

2.9.2. Replay protection

Security mechanisms can offer trade-offs between performance and the security properties freshness and replay protection. They sometimes use names such as "0RTT" (zero round-trip time). With those mechanisms, the server recognizes that a request is sent in such a 0RTT mode, but can still decide to send a response. The general trade-off is that an attacker may intercept such a message and replay it at any later time, possibly multiple times, without the server having a reliable way of recognizing the replay.

The semantics of CoAP are conducive to using such facilities: Safe requests are recognized by their request method code to not have side effects. Nonetheless, more aspects need to be considered: There is no risk of metadata revealing data if the server answers a request multiple times. Kinds of metadata to look out for are the size of the response (which, in a replay situation, can give an active attacker additional data) as well as any processing delays. (Side effects would also fall into that category, but there should not be any effects for safe requests).

If nothing else, GET requests to constant resources, such as queries to /.well-known/core, can often be responded to safely on the CoAP layer even without any replay protection.

There are resources for which more requests than those with safe request methods may be handled without replay protection, but as that assessment is hard to make, it is prudent to err at the side of caution.

CoAP has no error code like HTTP's 425 Too Early with which a server could ask the client to resubmit its request later when all the security mechanism's guarantees are available. Instead, servers can send 4.01 Unauthorized responses with Echo option; clients then repeat the request with the Echo value in the request.

Appendix B.1.2 of [RFC8613] describes how this is used to recover loss of state in OSCORE. Exceeding what is described there, a server can safely send a successful response, provided its criteria for 0RTT responses are met. The server can still send an Echo option with the successful response: Only when the client repeats that Echo value in a subsequent response can the replay window be initialized.

Implementers of OSCORE should be aware that answering potential replays can only be determined to be safe from the CoAP application's point of view. As always, unless the sequence number in the request has just been removed from an initialized replay window, the response can not reuse the request's nonce, but needs to be sent with a new sequence number from the server's space.

Requests with 0RTT properties currently cannot happen in DTLS because 0-RTT Data is not allowed for CoAP (cf. Section 14 of [I-D.ietf-uta-tls13-iot-profile]). However, that may change if a future document defines a profile for using early data with CoAP.

2.10. RFC 7252-12.3: Content-Format Registry

Section 12.3 of [RFC7252] established the CoAP Content-Formats Registry, which maps a combination of an Internet Media Type with an HTTP Content Coding, collectively called a Content-Format, to a concise numeric identifier for that Content-Format. The "Media Type" is more than a Media-Type-Name (see [RFC9193] for an extensive discussion), i.e., it may contain parameters beyond the mere combination of a type-name and a subtype-name registered in [IANA.media-types], as per [RFC6838], conventionally identified by the two names separated by a slash. This construct is often called a Content Type to reduce the confusion with a Media-Type-Name (e.g., in Section 8.3 of [RFC9110], which then however also opts to use the term Media Type for the same information set).

The second column of the Content-Format registry is the Content Coding, which is defined in Section 8.4.1 of [RFC9110]. For historical reasons, the HTTP header field that actually carries the content coding is called Content-Encoding; this often leads to the misnaming of Content Coding as "content encoding".

As has been noted in [Err4954], the text in Section 12.3 of [RFC7252] incorrectly uses these terms in the context of content types and content coding:

1. The field that describes the Content Type is called "Media Type". This can lead to the misunderstanding that this column just carries a Media-Type-Name (such as "text/plain"), while it actually also can carry media type parameters (as in "text/plain; charset=UTF-8").

2. The field that describes the Content Coding uses the incorrect name "Content Encoding".

INCORRECT, CORRECTED:
The VERIFIED Errata Report [Err4954] corrects the usage of "Content-Encoding" in the text and changes the name of the first column of the Content-Format registry to "Content Type" and the name of the second field to "Content Coding". This change has been carried out by IANA.

[Err4954] also has some notes on what would be valid or invalid Content-Format registrations. These are not repeated here; they are however quite useful as a reference when preparing additional Content-Format registrations.

3. IANA Considerations

This document makes no new requests to IANA.

Individual clarifications may contain IANA considerations; as for example in Section 2.10.

4. Security Considerations

This document provides a number of corrections and clarifications to existing RFCs, but it does not make any changes with regard to the security aspects of the protocol. As a consequence, the security considerations of the referenced RFCs apply without additions.

(To be changed when that is no longer true; probably the security considerations will then be on the individual clarifications.)

5. References

5.1. 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>.

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

[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>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/rfc/rfc7641>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/rfc/rfc7959>.

[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>.

[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>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/rfc/rfc8323>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[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>.

5.2. 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>.

[CF-MATCHER] "EndpointContextMatcher.java", <https://github.com/eclipse-californium/californium/blob/main/element-connector/src/main/java/org/eclipse/californium/elements/EndpointContextMatcher.java>.

[COAP-RESOURCE] "libcoap: coap_resource(3)", n.d., <https://libcoap.net/doc/reference/4.3.5/man_coap_resource.html>.

[Err4895] RFC Errata Report 4895, RFC 7252, <https://www.rfc-editor.org/errata/eid4895>.

[Err4954] RFC Errata Report 4954, RFC 7252, <https://www.rfc-editor.org/errata/eid4954>.

[Err5078] RFC Errata Report 5078, RFC 7252, <https://www.rfc-editor.org/errata/eid5078>.

[I-D.bormann-core-responses] Bormann, C. and C. Amsss, "CoAP: Non-traditional response forms", Work in Progress, Internet-Draft, draft-bormann-core-responses-06, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-bormann-core-responses-06>.

[I-D.ietf-core-conditional-attributes]
Silverajan, B., Koster, M., and A. Soloway, "Conditional Query Parameters for CoAP Observe", Work in Progress, Internet-Draft, draft-ietf-core-conditional-attributes-12, 16 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-conditional-attributes-12>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E. and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-18, 10 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-18>.

[I-D.ietf-core-oscore-capable-proxies]
Tiloca, M. and R. Hglund, "OSCORE-capable Proxies", Work in Progress, Internet-Draft, draft-ietf-core-oscore-capable-proxies-06, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-capable-proxies-06>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. Hglund, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-28, 23 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-28>.

[I-D.ietf-core-oscore-key-update]
Hglund, R. and M. Tiloca, "Key Update for OSCORE (KUDOS)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-update-13, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-update-13>.

[I-D.ietf-core-uri-path-abbrev]
Amsss, C. and M. Richardson, "URI-Path abbreviation in CoAP", Work in Progress, Internet-Draft, draft-ietf-core-uri-path-abbrev-03, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-uri-path-abbrev-03>.

[I-D.ietf-tls-dtls-rrc]
Tschofenig, H., Kraus, A., and T. Fossati, "Return Routability Check for DTLS 1.2 and DTLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls-rrc-20, 14 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls-rrc-20>.

[I-D.ietf-uta-tls13-iot-profile]
Tschofenig, H., Fossati, T., Richardson, M., and D. Migault, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-19, 20 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-19>.

[I-D.irtf-t2trg-amplification-attacks]
Mattsson, J. P., Selander, G., and C. Amsss, "Amplification Attacks Using the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-irtf-t2trg-amplification-attacks-05, 18 June 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-amplification-attacks-05>.

[I-D.irtf-t2trg-rest-iot]
Kernen, A., Kovatsch, M., and K. Hartke, "Guidance on RESTful Design for Internet of Things Systems", Work in Progress, Internet-Draft, draft-irtf-t2trg-rest-iot-17, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-rest-iot-17>.

[IANA.media-types]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[RFC6943] Thaler, D., Ed., "Issues in Identifier Comparison for Security Purposes", RFC 6943, DOI 10.17487/RFC6943, May 2013, <https://www.rfc-editor.org/rfc/rfc6943>.

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

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/rfc/rfc7967>.

[RFC8516] Keranen, A., ""Too Many Requests" Response Code for the Constrained Application Protocol", RFC 8516, DOI 10.17487/RFC8516, January 2019, <https://www.rfc-editor.org/rfc/rfc8516>.

[RFC9006] Gomez, C., Crowcroft, J., and M. Scharf, "TCP Usage Guidance in the Internet of Things (IoT)", RFC 9006, DOI 10.17487/RFC9006, March 2021, <https://www.rfc-editor.org/rfc/rfc9006>.

[RFC9110] 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/rfc/rfc9110>.

[RFC9146] Rescorla, E., Ed., Tschofenig, H., Ed., Fossati, T., and A. Kraus, "Connection Identifier for DTLS 1.2", RFC 9146, DOI 10.17487/RFC9146, March 2022, <https://www.rfc-editor.org/rfc/rfc9146>.

[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>.

[RFC9193] Kernen, A. and C. Bormann, "Sensor Measurement Lists (SenML) Fields for Indicating Data Value Content-Format", RFC 9193, DOI 10.17487/RFC9193, June 2022, <https://www.rfc-editor.org/rfc/rfc9193>.

Acknowledgements

The present document is modeled after RFC 4815 and the Internet-Drafts of the ROHC WG that led to it. Many thanks to the co-chairs of the ROHC WG and WG members that made this a worthwhile and successful experiment at the time.

Author’s Address

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