

CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

M. Tiloca
RISE AB
E. Dijk
IoTconsultancy.nl
November 02, 2020

Proxy Operations for CoAP Group Communication
draft-tiloca-core-groupcomm-proxy-02

Abstract

This document specifies the operations performed by a forward-proxy, when using the Constrained Application Protocol (CoAP) in group communication scenarios. Proxy operations involve the processing of individual responses from servers, as reply to a single request sent by the client over unicast to the proxy, and then distributed by the proxy over IP multicast to the servers. When receiving the different responses via the proxy, the client is able to distinguish them and their origin servers, by acquiring their addressing information.

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 May 6, 2021.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. The Multicast-Signaling Option	4
3. The Response-Forwarding Option	5
4. Requirements and Objectives	6
5. Protocol Description	7
5.1. Request Sending	7
5.1.1. Supporting Observe	9
5.2. Request Processing at the Proxy	9
5.2.1. Supporting Observe	10
5.3. Request and Response Processing at the Server	10
5.3.1. Supporting Observe	10
5.4. Response Processing at the Proxy	10
5.4.1. Supporting Observe	11
5.5. Response Processing at the Client	12
5.5.1. Supporting Observe	13
6. Chain of Proxies	13
6.1. Request Processing at the Proxy	14
6.1.1. Supporting Observe	15
6.2. Response Processing at the Proxy	16
6.2.1. Supporting Observe	16
7. Security Considerations	17
7.1. Client Authentication	17
7.2. Multicast-Signaling Option	18
7.3. Response-Forwarding Option	19
8. IANA Considerations	19
8.1. CoAP Option Numbers Registry	19
9. References	19
9.1. Normative References	20
9.2. Informative References	21
Appendix A. Using OSCORE Between Client and Proxy	21
A.1. Protecting the Request	22
A.2. Verifying the Request	22
A.3. Protecting the Response	23
A.4. Verifying the Response	23
Acknowledgments	23
Authors' Addresses	23

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] allows the presence of forward-proxies, as intermediary entities supporting clients to perform requests on their behalf.

CoAP supports also group communication over IP multicast [I-D.ietf-core-groupcomm-bis], where a group request can be addressed to multiple recipient servers, each of which may reply with an individual unicast response. As discussed in Section 2.3.3 of [I-D.ietf-core-groupcomm-bis], this group communication scenario poses a number of issues and limitations to proxy operations.

In particular, the client sends a single unicast request to the proxy, which the proxy forwards to a group of servers over IP multicast. Later on, the proxy delivers back to the client multiple responses to the original unicast request. As defined by [RFC7252], the multiple responses are delivered to the client inside separate CoAP messages, all matching (by Token) to the client's original unicast request. A possible alternative approach of performing aggregation of responses into a single CoAP response would require a specific aggregation content-format, which is not available yet. Both these approaches have open issues.

This specification considers the former approach, i.e. the proxy forwards the individual responses to a CoAP group request back to the client. The described method addresses all the related issues raised in Section 2.3.3 of [I-D.ietf-core-groupcomm-bis]. To this end, a dedicated signaling protocol is defined, using two new CoAP options.

In particular, the client explicitly confirms its support for receiving multiple responses to a proxied unicast request, i.e. one per origin server, and for how long it is willing to wait for responses. Also, when forwarding a response to the client, the proxy indicates the addressing information of the origin server. This enables the client to distinguish the multiple, different responses by origin and to possibly contact one or more of the servers by sending individual unicast requests, optionally bypassing the forward-proxy.

1.1. 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.

Readers are expected to be familiar with terms and concepts defined in CoAP [RFC7252], Group Communication for CoAP [I-D.ietf-core-groupcomm-bis], CBOR [I-D.ietf-cbor-7049bis], OSCORE [RFC8613] and Group OSCORE [I-D.ietf-core-oscore-groupcomm].

2. The Multicast-Signaling Option

The Multicast-Signaling Option defined in this section has the properties summarized in Figure 1, which extends Table 4 of [RFC7252].

Since the option is not Safe-to-Forward, the column "N" indicates a dash for "not applicable". The value of the Multicast-Signaling Option specifies a timeout value in seconds, encoded as an unsigned integer (see Section 3.2 of [RFC7252]).

No.	C	U	N	R	Name	Format	Length	Default
TBD1		x	-		Multicast-Signaling	uint	0-5	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Figure 1: The Multicast-Signaling Option.

This document specifically defines how this option is used by a client, to indicate to a forward-proxy its support for and interest in receiving multiple responses to a proxied CoAP group request, i.e. one per origin server, and for how long it is willing to wait for receiving responses via that proxy (see Section 5.1 and Section 5.2).

The client, when sending a CoAP group request to a proxy via IP unicast, to be forwarded by the proxy to a targeted group of servers, includes the Multicast-Signaling Option into the request. The option value indicates after what time period in seconds the client will stop accepting responses matching its original unicast request, with the exception of notifications if CoAP Observe is used [RFC7641]. This allows the intermediary proxy to stop forwarding responses back to the client, if received from the servers later than a timeout expiration.

The Multicast-Signaling Option is of class U in terms of OSCORE processing (see Section 4.1 of [RFC8613]).

3. The Response-Forwarding Option

The Response-Forwarding Option defined in this section has the properties summarized in Figure 2, which extends Table 4 of [RFC7252]. The option is intended only for CoAP responses, and builds on the Base-Uri option from Section 3 of [I-D.bormann-coap-misc].

Since the option is intended only for responses, the column "N" indicates a dash for "not applicable".

No.	C	U	N	R	Name	Format	Length	Default
TBD2			-		Response-Forwarding	(*)	9-24	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

(*) See below.

Figure 2: The Response-Forwarding Option.

This document specifically defines how this option is used by a proxy that forwards a request originated by a client over IP multicast.

Upon receiving a response to that request from a server, the proxy includes the Response-Forwarding Option into the response sent to the origin client (see Section 5). The proxy uses the option to indicate to the client the addressing information of the server generating the response.

The client can use the addressing information of the server specified in the option to identify the response originator and possibly send it individual requests later on, either directly or via the proxy as CoAP unicast requests.

The option value is the byte serialization of a CBOR array 'srv_info', which includes the following elements.

- o 'srv_addr': this element is a CBOR byte string, with value the unicast IP address of the server. This element is tagged and identified by the CBOR tag 260 "Network Address (IPv4 or IPv6 or MAC Address)". This element MUST be present.

- o 'srv_port': this element is a CBOR unsigned integer, with value the destination port number where to send unicast requests to the server. This element MAY be present.

The CDDL notation [RFC8610] provided below describes the 'srv_info' CBOR array using the format above.

```
srv_info = [
  srv_addr : #6.260(bstr), ; IP address of the server
  ? srv_port : uint, ; Port number of the server
]
```

If the 'srv_info' array does not include the element 'srv_port', it is assumed that the port number where to send unicast requests to the server is the same port number specified in the group URI of the original unicast CoAP group request sent to the proxy (see Section 5.1).

The Response-Forwarding Option is of class U in terms of OSCORE processing (see Section 4.1 of [RFC8613]).

4. Requirements and Objectives

This specification assumes that the following requirements are fulfilled.

- o REQ1. The CoAP proxy is explicitly configured (allow-list) to allow proxied CoAP group requests from specific client(s).
- o REQ2. The CoAP proxy MUST identify a client sending a CoAP group request, in order to verify whether the client is allowed-listed to do so. For example, this can rely on one of the following.
 - * A DTLS channel [RFC6347][I-D.ietf-tls-dtls13] between the client and the proxy, where the client has also been authenticated during the secure channel establishment.
 - * A pairwise OSCORE Security Context between the client and the proxy, as described in Appendix A.
- o REQ3. If secure, end-to-end communication is required between the client and the servers in the CoAP group, exchanged messages MUST be protected by using Group OSCORE [I-D.ietf-core-oscore-groupcomm], as discussed in Section 5.2 of [I-D.ietf-core-groupcomm-bis]. This requires the client and the servers to have previously joined the correct OSCORE group, for instance by using the approach described in [I-D.ietf-ace-key-groupcomm-oscore]. The correct OSCORE group to

join can be pre-configured or alternatively discovered, for instance by using the approach described in [I-D.tiloca-core-oscore-discovery].

This specification defines how to achieve the following objectives.

- o OBJ1. The CoAP proxy gets an indication from the client that it is in fact interested to and capable to receive multiple responses to its unicast request containing a CoAP group URI.
- o OBJ2. The CoAP proxy learns how long it should wait for responses to a proxied request, before starting to ignore following responses (except for notifications, if CoAP Observe is used [RFC7641]).
- o OBJ3. The CoAP proxy returns individual unicast responses to the client, each of which matches the original unicast request to the proxy.
- o OBJ4. The CoAP client is able to distinguish the different responses to the original unicast request, as well as their corresponding origin servers.
- o OBJ5. The CoAP client is enabled to optionally contact one or more of the responding origin servers in the future, either directly or via the CoAP proxy.

5. Protocol Description

This section specifies the steps of the signaling protocol.

5.1. Request Sending

In order to send a request addressed to a group of servers via the CoAP proxy, the client proceeds as follows.

1. The client prepares a request addressed to the CoAP proxy. The request specifies the group URI as a string in the Proxi-URI option, or by using the Proxy-Scheme option with the group URI constructed from the URI-* options (see Section 2.3.3 of [I-D.ietf-core-groupcomm-bis]).
2. The client MUST retain the Token value used for this original unicast request beyond the reception of a first response matching it. To this end, the client follows the same rules for Token retention defined for multicast requests in Section 2.3.1 of [I-D.ietf-core-groupcomm-bis].

In particular, the client picks an amount of time T it is fine to wait for before freeing up the Token value. Specifically, the value of T MUST be such that:

- * $T < T_r$, where T_r is the amount of time that the client is fine to wait for before potentially reusing the Token value. Note that T_r MUST NOT be less than `MIN_TOKEN_REUSE_TIME` defined in Section 2.3.1 of [I-D.ietf-core-groupcomm-bis].
 - * T should be at least the expected worst-case time taken by the request and response processing on the forward-proxy and on the servers in the addressed CoAP group.
 - * T should be at least the expected worst-case round-trip delay between the client and the forward-proxy, as well as between the proxy and the origin servers.
3. The client MUST include the Multicast-Signaling Option defined in Section 2 into the unicast request to send to the proxy. The option value specifies an amount of time $T' < T$. The difference ($T - T'$) should be at least the expected worst-case round-trip time between the client and the forward-proxy.
- The client can specify $T' = 0$ as option value, thus indicating to be not interested in receiving responses from the origin servers through the proxy. In such a case, the client SHOULD also include a No-Response Option [RFC7967] with value 26 (suppress all response codes), if it supports the option.
- Consistently, if the unicast request to send to the proxy already included a No-Response Option with value 26, the client SHOULD specify $T' = 0$ as value of the Multicast-Signaling Option.
4. The client processes the request as defined in [I-D.ietf-core-groupcomm-bis], and also as in [I-D.ietf-core-oscore-groupcomm] when secure group communication is used between the client and the servers.
5. If OSCORE is used to protect the leg between the client and the proxy (see REQ2 in Section 4), the client (further) protects the unicast request as resulting at the end of step 4. In particular, the client uses the pairwise OSCORE Security Context it has with the proxy, as described in Appendix A.1.
6. The client sends the request to the proxy as a unicast CoAP message.

The exact method that the client uses to estimate the worst-case processing times and round-trip delays mentioned above is out of the scope of this specification. However, such a method is expected to be already used by the client when generally determining a good Token lifetime and reuse interval.

5.1.1. Supporting Observe

When using CoAP Observe [RFC7641], the client follows what is specified in Section 2.3.5 of [I-D.ietf-core-groupcomm-bis], with the difference that it sends a unicast request to the proxy, to be forwarded to the group of servers, as defined in Section 5.1 of this specification.

Furthermore, the client especially follows what is specified in Section 5 of [RFC7641], i.e. it registers its interest to be an observer with the proxy, as if it was communicating with the servers.

5.2. Request Processing at the Proxy

Upon receiving the request from the client, the proxy proceeds as follows.

1. If OSCORE is used to protect the leg between the client and the proxy (see REQ2 in Section 4), the proxy decrypts the request using the pairwise OSCORE Security Context it has with the client, as described in Appendix A.2.
2. The proxy identifies the client, and verifies that the client is in fact allowed-listed to have its requests proxied to CoAP group URIs.
3. The proxy verifies the presence of the Multicast-Signaling Option, as a confirmation that the client is fine to receive multiple responses matching the same original request.

If the Multicast-Signaling Option is not present, the proxy MUST stop processing the request and MUST reply to the client with a 4.00 (Bad Request) response. The response MUST include a diagnostic payload, specifying that the Multicast-Signaling Option was missing and has to be included.

4. The proxy retrieves the value T' from the Multicast-Signaling Option, and then removes the option from the client's request.
5. The proxy forwards the client's request to the group of servers. In particular, the proxy sends it as a CoAP group request over IP multicast, addressed to the group URI specified by the client.

6. The proxy sets a timeout with the value T' retrieved from the Multicast-Signaling Option of the original unicast request.

In case $T' > 0$, the proxy will ignore responses to the forwarded group request coming from servers, if received after the timeout expiration, with the exception of Observe notifications (see Section 5.4).

In case $T' = 0$, the proxy will ignore all responses to the forwarded group request coming from servers.

5.2.1. Supporting Observe

When using CoAP Observe [RFC7641], the proxy takes the role of the client and registers its own interest to observe the target resource with the servers as per Section 5 of [RFC7641].

When doing so, the proxy especially follows what is specified for the client in Section 2.3.5 of [I-D.ietf-core-groupcomm-bis], by forwarding the group request to the servers over IP multicast, as defined in Section 5.2 of this specification.

5.3. Request and Response Processing at the Server

Upon receiving the group request from the proxy, a server proceeds as follows.

1. The server processes the group request as defined in [I-D.ietf-core-groupcomm-bis], and also as in [I-D.ietf-core-oscore-groupcomm] when secure group communication is used between the client and the server.
2. The server processes the response to be forwarded back to the client as defined in [I-D.ietf-core-groupcomm-bis], and also as in [I-D.ietf-core-oscore-groupcomm] when secure group communication is used between the client and the server.

5.3.1. Supporting Observe

When using CoAP Observe [RFC7641], the server especially follows what is specified in Section 2.3.5 of [I-D.ietf-core-groupcomm-bis] and Section 5 of [RFC7641].

5.4. Response Processing at the Proxy

Upon receiving a response matching the group request before the amount of time T' has elapsed, the proxy proceeds as follows.

1. The proxy MUST include the Response-Forwarding Option defined in Section 3 into the response. The proxy specifies as option value the addressing information of the server generating the response, encoded as defined in Section 3. In particular:
 - * The 'srv_addr' element of the 'srv_info' array MUST specify the server IPv6 address if the multicast request was destined for an IPv6 multicast address, and MUST specify the server IPv4 address if the multicast request was destined for an IPv4 address.
 - * If present, the 'srv_port' element of the 'srv_info' array MUST specify the port number of the server as the source port number of the response. This element MUST be present if the source port number of the response differs from the port number specified in the group URI of the original unicast CoAP group request (see Section 5.1). Otherwise, the 'srv_port' element MAY be omitted.
2. If OSCORE is used to protect the leg between the client and the proxy (see REQ2 in Section 4), the proxy (further) protects the response using the pairwise OSCORE Security Context it has with the client, as described in Appendix A.3.
3. The proxy forwards the response back to the client.

Upon timeout expiration, i.e. T' seconds after having sent the group request over IP multicast, the proxy frees up its local Token value associated to that request. Thus, following late responses to the same group request will be discarded and not forwarded back to the client.

5.4.1. Supporting Observe

When using CoAP Observe [RFC7641], the proxy acts as a client registered with the servers, as described earlier in Section 5.2.1.

Furthermore, the proxy takes the role of a server when forwarding notifications from origin servers back to the client. To this end, the proxy follows what is specified in Section 2.3.5 of [I-D.ietf-core-groupcomm-bis] and Section 5 of [RFC7641], with the following additions.

- o At step 1 in Section 5.4, the proxy includes the Response-Forwarding Option in every notification, including non-2.xx notifications resulting in removing the proxy from the list of observers of the origin server.

- o The proxy frees up its Token value used for a group observation only if, after the timeout expiration, no 2.xx (Success) responses matching the group request and also including an Observe option have been received from any origin server. After that, as long as observations are active with servers in the group for the target resource of the group request, notifications from those servers are forwarded back to the client, as defined in Section 5.4.

Finally, the proxy SHOULD regularly verify that the client is still interested in receiving observe notifications for a group observation. To this end, the proxy can rely on the same approach discussed for servers in Section 2.3.5 of [I-D.ietf-core-groupcomm-bis], with more details available in Section 4.5 of [RFC7641].

5.5. Response Processing at the Client

Upon receiving from the proxy a response matching the original unicast request before the amount of time T has elapsed, the client proceeds as follows.

1. The client processes the response as defined in [I-D.ietf-core-groupcomm-bis].
2. If OSCORE is used to protect the leg between the client and the proxy (see REQ2 in Section 4), the client decrypts the response using the pairwise OSCORE Security Context it has with the proxy, as described in Appendix A.4.
3. If secure group communication is used between the client and the servers, the client processes the response, possibly as outcome of step 2, as defined in [I-D.ietf-core-oscore-groupcomm].
4. The client identifies the origin server, whose addressing information is specified as value of the Response-Forwarding Option. If the port number is omitted in the value of the Response-Forwarding Option, the client MUST assume that the port number where to send unicast requests to the server is the same port number specified in the group URI of the original unicast CoAP group request sent to the proxy (see Section 5.1).

In particular, the client is able to distinguish different responses as originated by different servers. Optionally, the client may contact one or more of those servers individually, i.e. directly (bypassing the proxy) or indirectly (via a proxied CoAP unicast request).

Upon the timeout expiration, i.e. T seconds after having sent the original unicast request to the proxy, the client frees up its local Token value associated to that request. Note that, upon this timeout expiration, the Token value is not eligible for possible reuse yet (see Section 5.1). Thus, until the actual amount of time before enabling Token reuse has elapsed, following late responses to the same request forwarded by the proxy will be discarded, as not matching (by Token) any active request from the client.

5.5.1. Supporting Observe

When using CoAP Observe [RFC7641], the client frees up its Token value only if, after the timeout expiration, no 2.xx (Success) responses matching the original unicast request and also including an Observe option have been received.

Instead, if at least one such response has been received, the client continues receiving those notifications as forwarded by the proxy, as long as the observation for the target resource of the original unicast request is active.

6. Chain of Proxies

A client may be interested to access a resource at a group of origin servers which is reached through a chain of two or more proxies.

That is, these proxies are configured into a chain, where each non-last proxy is configured to forward CoAP (multicast) requests to the next hop towards the origin servers. Also, each non-first proxy is configured to forward back CoAP responses to (the previous hop proxy towards) the origin client.

This section specifies how the signaling protocol defined in Section 5 is used in that setting. Except for the last proxy before the origin servers, every other proxy in the chain takes the role of client with respect to the next hop towards the origin servers. Also, every proxy in the chain takes the role of server towards the previous proxy closer to the origin client.

The requirements REQ1 and REQ2 defined in Section 4 MUST be fulfilled for each proxy in the chain. That is, every proxy in the chain has to be explicitly configured (allow-list) to allow proxied group requests from specific senders, and MUST identify those senders upon receiving their group request. For the first proxy in the chain, that sender is the origin client. For each other proxy in the chain, that sender is the previous hop proxy closer the origin client. In either case, a proxy can identify the sender of a group request by the same means mentioned in Section 4.

6.1. Request Processing at the Proxy

Upon receiving a group request to be forwarded to a CoAP group URIs, a proxy proceed as follows.

If the proxy is the last one in the chain, i.e. it is the last hop before the origin servers, the proxy performs the steps defined in Section 5.2, with no modifications.

Otherwise, the proxy performs the steps defined in Section 5.2, with the following differences.

- o At steps 1-3, "client" refers to the origin client for the first proxy in the chain; or to the previous hop proxy closer to the origin client, otherwise.
- o At step 4, the proxy rather performs the following actions.
 1. The proxy retrieves the value T' from the Multicast-Signaling Option, and does not remove the option.
 2. In case $T' > 0$, the proxy picks an amount of time T it is fine to wait for before freeing up its local Token value to use with the next hop towards the origin servers. To this end, the proxy MUST follow what defined at step 2 of Section 5.1 for the origin client, with the following differences.
 - + T MUST be greater than the retrieved value T' , i.e. $T' < T$.
 - + The worst-case message processing time takes into account all the next hops towards the origin servers, as well as the origin servers themselves.
 - + The worst-case round-trip delay takes into account all the legs between the proxy and the origin servers.
 3. In case $T' > 0$, the proxy replaces the value of the Multicast-Signaling Option with a new value T'' , such that:
 - + $T'' < T$. The difference $(T - T'')$ should be at least the expected worst-case round-trip time between the proxy and the next hop towards the origin servers.
 - + $T'' < T'$. The difference $(T' - T'')$ should be at least the expected worst-case round-trip time between the proxy and the (previous hop proxy closer to the) origin client.

If the proxy is not able to determine a value T'' that fulfills both the requirements above, the proxy MUST stop processing the request and MUST respond with a 5.05 (Proxying Not Supported) error response to the previous hop proxy closer to the origin client. The proxy SHOULD include a Multicast-Signaling Option, set to the minimum value T' that would be acceptable in the Multicast-Signaling Option of a request to forward.

Upon receiving such an error response, any proxy in the chain MAY send an updated request to the next hop towards the origin servers, specifying in the Multicast-Signaling Option a value T' greater than in the previous request. If this does not happen, the proxy receiving the error response MUST also send a 5.05 (Proxying Not Supported) error response to the previous hop proxy closer to the origin client. Like the received one, also this error response SHOULD include a Multicast-Signaling Option, set to the minimum value T' acceptable by the proxy sending the error response.

- o At step 5, the proxy forwards the request to the next hop towards the origin servers.
- o At step 6, the proxy sets a timeout with the value T' retrieved from the Multicast-Signaling Option of the request received from the (previous hop proxy closer to the) origin client.

In case $T' > 0$, the proxy will ignore responses to the forwarded group request coming from the (next hop towards the) origin servers, if received after the timeout expiration, with the exception of Observe notifications (see Section 5.4).

In case $T' = 0$, the proxy will ignore all responses to the forwarded group request coming from the (next hop towards the) origin servers.

6.1.1. Supporting Observe

When using CoAP Observe [RFC7641], what defined in Section 5.2.1 applies for the last proxy in the chain, i.e. the last hop before the origin servers.

Any other proxy in the chain acts as a client and registers its own interest to observe the target resource with the next hop towards the origin servers, as per Section 5 of [RFC7641].

6.2. Response Processing at the Proxy

Upon receiving a response matching the group request before the amount of time T' has elapsed, the proxy proceeds as follows.

If the proxy is the last one in the chain, i.e. it is the last hop before the origin servers, the proxy performs the steps defined in Section 5.4, with no modifications.

Otherwise, the proxy performs the steps defined in Section 5.4, with the following differences.

- o The proxy skips step 1. In particular, the proxy **MUST NOT** remove, alter or replace the Response-Forwarding Option.
- o At steps 2-3, "client" refers to the origin client for the first proxy in the chain; or to the previous hop proxy closer to the origin client, otherwise.

Upon timeout expiration, i.e. T seconds after having sent the group request to the next hop towards the origin servers, the proxy frees up its local Token value associated to that request. Thus, following late responses to the same group request will be discarded and not forwarded back to the (previous hop proxy closer to the) origin client.

6.2.1. Supporting Observe

When using CoAP Observe [RFC7641], what defined in Section 5.4.1 applies for the last proxy in the chain, i.e. the last hop before the origin servers.

As to any other proxy in the chain, the following applies.

- o The proxy acts as a client registered with the next hop towards the origin servers, as described earlier in Section 6.1.1.
- o The proxy takes the role of a server when forwarding notifications from the next hop to the origin servers back to the (previous hop proxy closer to the) origin client, as per Section 5 of [RFC7641].
- o The proxy frees up its Token value used for a group observation only if, after the timeout expiration, no 2.xx (Success) responses matching the group request and also including an Observe option have been received from the next hop towards the origin servers. After that, as long as the observation for the target resource of the group request is active with the next hop towards the origin servers in the group, notifications from that hop are forwarded

back to the (previous hop proxy closer to the) origin client, as defined in Section 6.2.

- o The proxy SHOULD regularly verify that the (previous hop proxy closer to the) origin client is still interested in receiving observe notifications for a group observation. To this end, the proxy can rely on the same approach defined in Section 4.5 of [RFC7641].

7. Security Considerations

The security considerations from [RFC7252][I-D.ietf-core-groupcomm-bis][RFC8613][I-D.ietf-core-oscore-groupcomm] hold for this document.

When a chain of proxies is used (see Section 6), the secure communication between any two adjacent hops is independent.

When Group OSCORE is used for end-to-end secure group communication between the origin client and the origin servers, this security association is unaffected by the possible presence of a proxy or a chain of proxies.

Furthermore, the following additional considerations hold.

7.1. Client Authentication

As per the requirement REQ2 (see Section 4), the client has to authenticate to the proxy when sending a group request to forward. This leverages an established security association between the client and the proxy, that the client uses to protect the group request, before sending it to the proxy.

Note that, if the group request is (also) protected with Group OSCORE, i.e. end-to-end between the client and the servers, the proxy can authenticate the client by successfully verifying the counter signature embedded in the group request. This requires that, for each client to authenticate, the proxy stores the public key used by that client in the OSCORE group, which in turn would require a form of active synchronization between the proxy and the Group Manager for that group [I-D.ietf-core-oscore-groupcomm].

Nevertheless, the client and the proxy SHOULD still rely on a full-fledged, pairwise secure association. In addition to ensuring the integrity of group requests sent to the proxy (see Section 7.2 and Section 7.3), this prevents the proxy from forwarding replayed group requests with a valid counter signature, as possibly injected by an active, on-path adversary.

The same considerations apply when a chain of proxies is used (see Section 6), with each proxy but the last one in the chain acting as client with the next hop towards the origin servers.

7.2. Multicast-Signaling Option

The Multicast-Signaling Option is of class U for OSCORE [RFC8613]. Hence, also when Group OSCORE is used between the client and the servers [I-D.ietf-core-oscore-groupcomm], a proxy is able to access the option value and retrieve the timeout value T' , as well as to remove the option altogether before forwarding the group request to the servers. When a chain of proxies is used (see Section 6), this also allows each proxy but the last one in the chain to update the option value, as an indication for the next hop towards the origin servers (see Section 6.1).

The security association between the client and the proxy **MUST** provide message integrity, so that further intermediaries between the two as well as on-path active adversaries are not able to remove the option or alter its content, before the group request reaches the proxy. Removing the option would otherwise result in not forwarding the group request to the servers. Instead, altering the option content would result in the proxy accepting and forwarding back responses for an amount of time different than the one actually indicated by the client.

The security association between the client and the proxy **SHOULD** also provide message confidentiality. Otherwise, further intermediaries between the two as well as on-path passive adversaries would be able to simply access the option content, and thus learn for how long the client is willing to receive responses from the servers in the group via the proxy. This may in turn be used to perform a more efficient, selective suppression of responses from the servers.

When the client (further) protects the unicast request sent to the proxy using OSCORE (see Appendix A) and/or with DTLS, both message integrity and message confidentiality are achieved in the leg between the client and the proxy.

The same considerations above about security associations apply when a chain of proxies is used (see Section 6), with each proxy but the last one in the chain acting as client with the next hop towards the origin servers.

7.3. Response-Forwarding Option

The Response-Forwarding Option is of class U for OSCORE [RFC8613]. Hence, also when Group OSCORE is used between the client and the servers [I-D.ietf-core-oscore-groupcomm], the proxy that has forwarded the group request to the servers is able to include the option into a server response, before forwarding this response back to the (previous hop proxy closer to the) origin client.

Since the security association between the client and the proxy provides message integrity, any further intermediaries between the two or on-path active adversaries are not able to undetectably remove the Response-Forwarding Option from a forwarded server response. This ensures that the client can correctly distinguish the different responses and identify their corresponding origin server.

When the proxy (further) protects the response forwarded back to the client using OSCORE (see Appendix A) and/or with DTLS, message integrity is achieved in the leg between the client and the proxy.

The same considerations above about security associations apply when a chain of proxies is used (see Section 6), with each proxy but the last one in the chain acting as client with the next hop towards the origin servers.

8. IANA Considerations

This document has the following actions for IANA.

8.1. CoAP Option Numbers Registry

IANA is asked to enter the following option numbers to the "CoAP Option Numbers" registry defined in [RFC7252] within the "CoRE Parameters" registry.

Number	Name	Reference
TBD1	Multicast-Signaling	[[this document]]
TBD2	Response-Forwarding	[[this document]]

9. References

9.1. Normative References

- [I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", draft-ietf-cbor-7049bis-16 (work in progress), September 2020.
- [I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", draft-ietf-core-groupcomm-bis-02 (work in progress), November 2020.
- [I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", draft-ietf-core-oscore-groupcomm-10 (work in progress), November 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <<https://www.rfc-editor.org/info/rfc7641>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [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/info/rfc8613>>.

9.2. Informative References

- [I-D.bormann-coap-misc]
Bormann, C. and K. Hartke, "Miscellaneous additions to CoAP", draft-bormann-coap-misc-27 (work in progress), November 2014.
- [I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", draft-ietf-ace-key-groupcomm-oscore-09 (work in progress), November 2020.
- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-38 (work in progress), May 2020.
- [I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", draft-tiloca-core-oscore-discovery-07 (work in progress), November 2020.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [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/info/rfc7967>>.

Appendix A. Using OSCORE Between Client and Proxy

This section describes how OSCORE is used to protect messages exchanged by an origin client and a proxy, using their pairwise OSCORE Security Context.

This is especially convenient for the communication scenario addressed in this document, when the origin client already supports and uses Group OSCORE [I-D.ietf-core-oscore-groupcomm] to protect messages end-to-end with the origin servers.

The following focuses on the origin client originating the group request and a single proxy as its immediate next hop. When a chain of proxies is used (see Section 6), the same independently applies between each pair of proxies in the chain, where the proxy forwarding

the group request acts as client and the next hop towards the origin servers acts as server.

A.1. Protecting the Request

Before sending the CoAP request to the proxy, the origin client protects it using the pairwise OSCORE Security Context it has with the proxy.

To this end, the origin client processes the CoAP request as defined in Section 8.1 of [RFC8613], with the following differences.

- o The Proxy-Uri option, if present, is not decomposed and recomposed as defined in Section 4.1.3.3 of [RFC8613].
- o The following options, if present, are processed as Class E.
 - * Proxy-Uri, Proxy-Scheme, Uri-Host and Uri-Port, defined in [RFC7252].
 - * OSCORE, defined in [RFC8613], which is present if Group OSCORE is used between the origin client and the origin servers, to achieve end-to-end secure group communication.
 - * Multicast-Signaling Option, defined in Section 2 of this specification.

As per [RFC8613], the resulting message includes an outer OSCORE Option, which reflects the usage of the pairwise OSCORE Security Context between the origin client and the proxy.

A.2. Verifying the Request

The proxy verifies the CoAP request as defined in Section 8.2 of [RFC8613]. Note that the Multicast-Signaling Option is retrieved during the decryption process, and added to the decrypted request.

If secure group communication is also used between the origin client and the origin servers, the request resulting from the previous step and to be forwarded to the origin servers is also already protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm]. Consequently, it includes an outer OSCORE Option, which reflects the usage of the group OSCORE Security Context between the origin client and the origin servers.

A.3. Protecting the Response

The proxy protects the CoAP response received from a server, using the pairwise OSCORE Security Context it has with the origin client.

To this end, the proxy processes the CoAP response as defined in Section 8.3 of [RFC8613], with the difference that the OSCORE Option, if present, is processed as Class E. This is the case if Group OSCORE is used between the origin client and the origin servers, to achieve end-to-end secure group communication.

Furthermore, the Response-Forwarding Option defined in Section 3 of this specification is also processed as Class E.

As per [RFC8613], the resulting message to be forwarded back to the origin client includes an outer OSCORE Option, which reflects the usage of the pairwise OSCORE Security Context between the origin client and the proxy.

A.4. Verifying the Response

The origin client verifies the CoAP response received from the proxy as defined in Section 8.4 of [RFC8613]. Note that, the Response-Forwarding Option is retrieved during the decryption process, and added to the decrypted response.

If secure group communication is also used between the origin client and the origin servers, the response resulting from the previous step is protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm]. Consequently, it includes an outer OSCORE Option, which reflects the usage of the group OSCORE Security Context between the origin client and the origin servers.

Acknowledgments

The authors sincerely thank Christian Amsuess, Jim Schaad and Goeran Selander for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: marco.tiloca@ri.se

Esko Dijk
IoTconsultancy.nl
_____\n
Utrecht
The Netherlands

Email: esko.dijk@iotconsultancy.nl