CoRE Working Group
Internet-Draft
Intended status: Informational
Expires: 24 October 2026

M. Tiloca
R. H旦glund
RISE AB
C. Ams端ss

F. Palombini
Ericsson AB

22 April 2026

Using Proxies for Observe Notifications as CoAP Multicast Responses
draft-ietf-core-multicast-notifications-proxy-01

Abstract

The Constrained Application Protocol (CoAP) allows clients to "observe" resources at a server and to receive notifications as unicast responses upon changes of the resource state. Instead of sending a distinct unicast notification to each different client, a server can alternatively send a single notification as a response message over multicast, to all the clients observing the same target resource. When doing so, the security protocol Group Object Security for Constrained RESTful Environments (Group OSCORE) can be used to protect multicast notifications end-to-end between the server and the observer clients. This document describes how multicast notifications can be used in network setups that leverage a proxy, e.g., in order to accommodate clients that are not able to directly listen to multicast traffic.

Discussion Venues

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

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

Source for this draft and an issue tracker can be found at https://github.com/core-wg/multicast-notifications-proxy.

Status of This Memo

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

This Internet-Draft will expire on 24 October 2026.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] has been extended with a number of mechanisms, including resource Observation [RFC7641]. This enables CoAP clients to register at a CoAP server as "observers" of a resource, and hence being automatically notified with an unsolicited response upon changes of the resource state.

CoAP supports group communication [I-D.ietf-core-groupcomm-bis], e.g., over IP multicast. This includes support for Observe registration requests over multicast, in order for clients to efficiently register as observers of a resource hosted at multiple servers.

In a number of use cases, it is conversely desirable that a server sends observe notifications for the same target resource to multiple observers at once. In general, this is beneficial when several CoAP clients observe the same target resource at a CoAP server, and thus they could all be notified at once by means of a single response message.

To this end, [I-D.ietf-core-observe-multicast-notifications] defines a method that a server can use to deliver observe notifications as CoAP responses addressed to multiple clients, e.g., over IP multicast. Also, it defines how to use the security protocol Group Object Security for Constrained RESTful Environments (Group OSCORE) [I-D.ietf-core-oscore-groupcomm] to protect multicast notifications end-to-end between the server and the observer clients.

This document describes how the method specified in [I-D.ietf-core-observe-multicast-notifications] can be used in network setups that leverage a proxy, e.g., in order to accommodate clients that are not able to directly listen to multicast traffic.

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 described in CoAP [RFC7252], group communication for CoAP [I-D.ietf-core-groupcomm-bis], Observe [RFC7641], Concise Data Definition Language (CDDL) [RFC8610], Concise Binary Object Representation (CBOR) [RFC8949], Object Security for Constrained RESTful Environments (OSCORE) [RFC8613], Group OSCORE [I-D.ietf-core-oscore-groupcomm], and Constrained Resource Identifiers (CRIs) [I-D.ietf-core-href].

Readers are also expected to be familiar with terms and concepts described in [I-D.ietf-core-observe-multicast-notifications], particularly with the terms "traditional observation", "group observation", "phantom request", and "informative response".

2. High-Level Overview of Available Variants

Building on what is specified in [I-D.ietf-core-observe-multicast-notifications], this document considers network setups where proxies are deployed, which is expected if (some of) the clients participating in the group observation are not capable to listen to multicast traffic. In such setups, a proxy directly receives multicast notifications from the server and relays them back to the clients.

Therefore, with respect to [I-D.ietf-core-observe-multicast-notifications], this document introduces additional variants to enforce a group observation. As a complement to Section 3 of [I-D.ietf-core-observe-multicast-notifications], the rest of this section provides an overview of such additional variants, which differ as to whether exchanged messages are protected end-to-end between the observer clients and the server.

* Variant with proxy and without end-to-end security - Messages pertaining to the group observation are not protected end-to-end between the clients and the server. This basic case is defined in Section 3. An example is provided in Section 6.

* Variant with proxy and with end-to-end security - Messages pertaining to the group observation are protected end-to-end between the clients and the server, by using the security protocol Group OSCORE [I-D.ietf-core-oscore-groupcomm]. In particular, the clients separately provide the proxy with the obtained phantom request, thus enabling the proxy to receive the multicast notifications from the server. This case is defined in Section 4. An example is provided in Section 7.

  If the participating endpoints using Group OSCORE also support the concept of Deterministic Client [I-D.ietf-core-cacheable-oscore], the same advantages mentioned in Section 3 of [I-D.ietf-core-observe-multicast-notifications] for the case without a proxy apply (see Appendix D of [I-D.ietf-core-observe-multicast-notifications]). In addition, this allows for a more efficient setup and enforcement of the group observation, by reducing the amount of message exchanges and allowing the proxy to effectively serve protected multicast notifications from its cache. An example is provided in Section 8.2.

3. Setup with Proxies

This section specifies how the approach presented in Sections 4 and 5 of [I-D.ietf-core-observe-multicast-notifications] works when a proxy is used between the clients and the server. In addition to what is specified in Section 5.7 of [RFC7252] and Section 5 of [RFC7641], the following applies.

A client sends its original observation request to the proxy. If the proxy is not already registered at the server for that target resource, the proxy forwards the observation request to the server, hence registering itself as an observer. If the server has an ongoing group observation for the target resource or decides to start one, the server considers the proxy as taking part in the group observation and replies to the proxy with an informative response see Section 4.2 of [I-D.ietf-core-observe-multicast-notifications].

Upon receiving an informative response, the proxy performs as specified for the client in Section 5 of [I-D.ietf-core-observe-multicast-notifications], with the peculiarity that "consuming" the last notification (if present) means populating the proxy’s cache.

In particular, by using the information retrieved from the informative response, the proxy configures an observation of the target resource at the origin server, acting as a client directly taking part in the group observation.

As a consequence, the proxy listens to the IP multicast address and port number indicated by the server, i.e., per the CRI [I-D.ietf-core-href] specified by a dedicated element of 'tpi_details' within the 'tp_info' parameter, in the informative response. In particular, when transporting CoAP over UDP, the CRI is conveyed by the element 'tpi_client' (see Section 4.2.1.1 of [I-D.ietf-core-observe-multicast-notifications]).

Furthermore, multicast notifications will match the phantom request stored at the proxy, based on the Token value specified by a dedicated element of 'tpi_details' within the 'tp_info' parameter, in the informative response. In particular, when transporting CoAP over UDP, the Token value is specified by the element 'tpi_token' (see Section 4.2.1.1 of [I-D.ietf-core-observe-multicast-notifications]).

Then, the proxy performs the following actions.

* If the 'last_notif' field is not present, the proxy replies to the client with an Empty Acknowledgement (if elicited by the message type of the original observation request, and if the proxy has not already done so).

* If the 'last_notif' field is present, the proxy rebuilds the latest multicast notification, as defined in Section 5 of [I-D.ietf-core-observe-multicast-notifications]. Then, the proxy replies to the client, by forwarding back the latest multicast notification.

When responding to an observation request from a client, the proxy also adds that client (and its Token) to the list of its registered observers for the target resource, next to the older observations.

Upon receiving a multicast notification from the server, the proxy forwards it back separately to each observer client over unicast. Note that the notification forwarded back to a certain client has the same Token value of the original observation request sent by that client to the proxy.

Note that the proxy configures the observation of the target resource at the server only once, when receiving the informative response associated with a (newly started) group observation for that target resource.

After that, when receiving an observation request from a following new client to be added to the same group observation, the proxy does not take any further action with the server. Instead, the proxy replies to the client either with the latest multicast notification if available from its cache, or with an Empty Acknowledgement otherwise, as defined above.

As a result, the observer counter at the server (see Section 4 of [I-D.ietf-core-observe-multicast-notifications]) is not incremented when a new origin client behind the proxy registers as an observer at the proxy. Instead, the observer counter takes into account only the proxy, which has registered as an observer at the server and has received the informative response from the server.

An example is provided in Section 6.

In the general case with a chain of two or more proxies, every proxy in the chain takes the role of client with the (next hop towards the) origin server. Note that the proxy adjacent to the origin server is the only one in the chain that receives informative responses and that listens to an IP multicast address and port number to receive notifications for the group observation. Furthermore, every proxy in the chain takes the role of server with the (previous hop towards the) origin client.

4. Setup with Proxies and with Group OSCORE

As defined in Section 9 of [I-D.ietf-core-observe-multicast-notifications], the security protocol Group OSCORE [I-D.ietf-core-oscore-groupcomm] can be used to protect multicast notifications end-to-end between the origin server and the origin clients.

Since the informative responses from the origin server are protected specifically end-to-end by using OSCORE or Group OSCORE, additional actions are required in the presence of a proxy.

In fact, the proxy adjacent to the origin server is not able to access the encrypted payload of such informative responses. Hence, the proxy cannot retrieve the ’ph_req’ and ’tp_info’ parameters necessary to correctly receive multicast notifications and forward them back to the clients.

Consequently, differently from what is defined in Section 11 of [I-D.ietf-core-observe-multicast-notifications], a proxy that receives an informative response simply forwards it back to the (previous hop towards the) origin client that has sent the corresponding observation request. Note that the proxy does not even realize that the message is an informative response, since the outer Code field is set to 2.05 (Content).

Upon receiving the informative response, the origin client does not configure an observation of the target resource yet. Instead, the origin client performs a new observe registration request, by transmitting the re-built phantom request as intended to reach the proxy adjacent to the origin server. In particular, the origin client includes the new CoAP option Listen-To-Multicast-Responses defined in Section 4.1, to provide that proxy with the transport-specific information required for receiving multicast notifications for the group observation.

As a result, the observer counter at the server (see Section 4 of [I-D.ietf-core-observe-multicast-notifications]) is incremented when, after having received the original observation request from a new origin client, the origin server replies with the informative response. In particular, the observer counter at the server reliably takes into account new, different origin clients behind the proxy, which the server distinguishes through their security identity specified by the pair (OSCORE Sender ID, OSCORE ID Context) in the OSCORE Option value of their original observation request. Note that this does not hold anymore if the origin endpoints use phantom observation requests as Deterministic Requests (see Appendix D of [I-D.ietf-core-observe-multicast-notifications]).

Details on the additional message exchange and processing are defined in Section 4.2.

4.1. Listen-To-Multicast-Responses Option

In order to allow a proxy to listen to multicast notifications sent by a server, a new CoAP option is defined. This option MUST be supported by clients that are interested to take part in group observations through intermediaries and by proxies that collect multicast notifications and forward them back to the observer clients.

The option is called Listen-To-Multicast-Response, is intended only for requests, and has the properties summarized in Table 1, which extends Table 4 of [RFC7252]. The option is critical and not Safe-to-Forward. Since the option is not Safe-to-Forward, the ’N’ column indicates a dash for "not applicable".

No.	C	U	N	R	Name	Format	Length	Default
TBD47	x	x	-		Listen-To-Multicast-Responses	(*)	3-1024	(none)

Table 1: The Listen-To-Multicast-Responses Option. C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Note to RFC Editor: In the table above, please replace TBD47 with the registered option number. Then, please delete this paragraph.

The value of the Listen-To-Multicast-Responses Option is the byte serialization of a CBOR array. The content of the array specifies transport-specific message information that is required for listening to the multicast notifications of a group observation and is intended to the proxy adjacent to the origin server sending those notifications. In particular, the serialized CBOR array has the same format specified in Section 4.2.1 of [I-D.ietf-core-observe-multicast-notifications] for the 'tp_info' parameter of the informative response defined in Section 4.2 of [I-D.ietf-core-observe-multicast-notifications].

The Listen-To-Multicast-Responses Option is of class U for OSCORE [RFC8613][I-D.ietf-core-oscore-groupcomm].

4.2. Message Processing

Compared to Section 3, the following additions apply when informative responses are protected end-to-end with Group OSCORE between the origin server and the origin clients.

After the origin server sends an informative response, each proxy simply forwards it back to the (previous hop towards the) origin client that has sent the observation request.

Once received the informative response, the origin client proceeds in a different way than the one defined in Section 9.3.1 of [I-D.ietf-core-observe-multicast-notifications]:

* The client performs all the additional decryption and verification steps of Section 9.3.1 of [I-D.ietf-core-observe-multicast-notifications] on the phantom request specified in the 'ph_req' parameter and on the last notification specified in the 'last_notif' parameter (if present).

* The client builds a ticket request (see Appendix C of [I-D.ietf-core-cacheable-oscore]), as intended to reach the proxy adjacent to the origin server. The ticket request is formatted as follows.

  - The Token is chosen as the client sees fit. In fact, there is no reason for this Token to be the same as the phantom request's.

  - The outer Code field, the outer CoAP options, and the encrypted payload (protecting the inner Code, the inner CoAP options, and the possible plain CoAP payload) concatenated with the countersignature are the same as those of the phantom request used for the group observation. That is, they are as specified in the 'ph_req' parameter of the received informative response.

  - An outer Observe Option is included and set to 0 (register). This is meant to be set in the phantom request already.

  - The client includes: the outer option Proxy-Uri or Proxy-Cri [I-D.ietf-core-href]; or the outer options (Uri-Host, Uri-Port), together with the outer option Proxy-Scheme or Proxy-Scheme-Number [I-D.ietf-core-href]. These options are set in order to specify the same request URI of the original registration request sent by the client.

  - The new option Listen-To-Multicast-Responses is included as an outer option. The value of the option is set to the byte serialization of the CBOR array specified by the 'tp_info' parameter of the informative response.

    Note that, except for transport-specific information such as the Token and Message ID values, every different client participating in the same group observation (hence rebuilding the same phantom request) will build the same ticket request.

    Note also that, identically to the phantom request, the ticket request is still protected with Group OSCORE, i.e., it has the same OSCORE Option, encrypted payload, and countersignature.

Then, the client sends the ticket request to the next hop towards the origin server. Every proxy in the chain forwards the ticket request to the next hop towards the origin server, until the last proxy in the chain is reached. This last proxy, adjacent to the origin server, proceeds as follows.

* The proxy MUST NOT further forward the ticket request to the origin server.

* The proxy removes the option Proxy-Uri, or Proxy-Scheme, or Proxy-Cri, or Proxy-Scheme-Number from the ticket request.

* The proxy removes the Listen-To-Multicast-Responses Option from the ticket request and extracts the transport-specific information specified within the option value.

* The proxy rebuilds the phantom request associated with the group observation, by using the ticket request as directly providing the required transport-independent information. This includes the outer Code field, the outer CoAP options, and the encrypted payload concatenated with the countersignature.

* The proxy configures an observation of the target resource at the origin server, acting as a client directly taking part in the group observation. To this end, the proxy uses the rebuilt phantom request and the transport-specific information retrieved from the Listen-To-Multicast-Responses Option. The particular way to achieve this is implementation specific.

After that, the proxy listens to the IP multicast address and port number indicated in the Listen-To-Multicast-Responses Option, i.e., per the CRI specified by a dedicated element of 'tpi_details' within the serialized CBOR array conveyed in the option value. In particular, when transporting CoAP over UDP, the CRI is conveyed by the element 'tpi_client' (see Section 4.2.1.1 of [I-D.ietf-core-observe-multicast-notifications]).

The multicast notifications of the group observation in question will match the phantom request stored at the proxy, based on the Token value specified by a dedicated element of 'tpi_details' within the serialized CBOR array conveyed in the value of the Listen-To-Multicast-Responses Option. In particular, when transporting CoAP over UDP, the Token value is specified by the element 'tpi_token' (see Section 4.2.1.1 of [I-D.ietf-core-observe-multicast-notifications]).

An example is provided in Section 7.

5. Impact from Proxies on Rough Counting of Clients in the Group Observation

Section 3 specifies how the approach defined in Sections 4 and 5 of [I-D.ietf-core-observe-multicast-notifications] works when a proxy is used between the origin clients and the origin server.

That is, the clients register as observers at the proxy, which in turn registers as a participant to the group observation at the server, receives the multicast notifications from the server, and forwards those to the clients.

With reference to the method defined in Section 8 of [I-D.ietf-core-observe-multicast-notifications], this has an impact on the rough counting that the server performs to keep an estimate of still active and interested clients. In particular, the following applies.

* Since the Feedback-Divider Option defined in Section 8.1 of [I-D.ietf-core-observe-multicast-notifications] is not Safe-to-Forward, the proxy needs to recognize and understand the option in order to participate to the rough counting process.

  If the proxy receives a request that includes the Feedback-Divider Option but the proxy does not recognize and understand the option, then the proxy stops processing the request and sends a 4.02 (Bad Option) response to the observer client (see Section 5.7.1 of [RFC7252]). This results in the client terminating its observation at the proxy, after which the client stops receiving notifications for the group observation.

  If the proxy receives a multicast notification that includes the Feedback-Divider Option but the proxy does not recognize and understand the option, then the proxy stops processing the received multicast notification and sends a 5.02 (Bad Gateway) response to each of the observer clients (see Section 5.7.1 of [RFC7252]). This results in all the observer clients terminating their observation at the proxy, after which they stop receiving notifications for the group observation. Consequently, the proxy may decide to forget about its participation to the group observation at the server.

  This is not an issue if communications between the origin endpoints are protected end-to-end, i.e., both for the requests from the origin clients by using OSCORE or Group OSCORE, as well as for the multicast notifications from the origin server by using Group OSCORE (see Section 9 of [I-D.ietf-core-observe-multicast-notifications] and Section 4 of the present document). In fact, in such a case, the Feedback-Divider Option is protected end-to-end as well, and is thus hidden from the proxy.

  Therefore, if the server uses the rough counting process defined in Section 8 of [I-D.ietf-core-observe-multicast-notifications] but communications are not protected end-to-end between the origin

endpoints, then it is practically required that the proxy recognizes and understands the Feedback-Divider Option. If that is not the case, then every execution of the rough counting process will effectively prevent the clients from receiving further notifications for the group observation, until they register again as observers at the proxy.

* The following holds when the proxy receives a multicast notification including the Feedback-Divider Option.

  - If the multicast notification is not protected end-to-end by using Group OSCORE (see Section 3), then the Feedback-Divider Option is visible to the proxy.

    In this case, the proxy proceeds like defined in Section 8.2 of [I-D.ietf-core-observe-multicast-notifications] for an origin client, i.e., by answering on its own to the server if it picks a random number I equal to 0. When doing so, the proxy will be counted by the server as a single client.

    Furthermore, the proxy MUST remove the option before forwarding the notification to (the previous hop towards) any of the origin clients.

    The proxy would have to rely on separate means for verifying whether the origin clients are still interested in the observation, e.g., by regularly forwarding notifications to the clients as unicast separate responses that are specifically Confirmable messages.

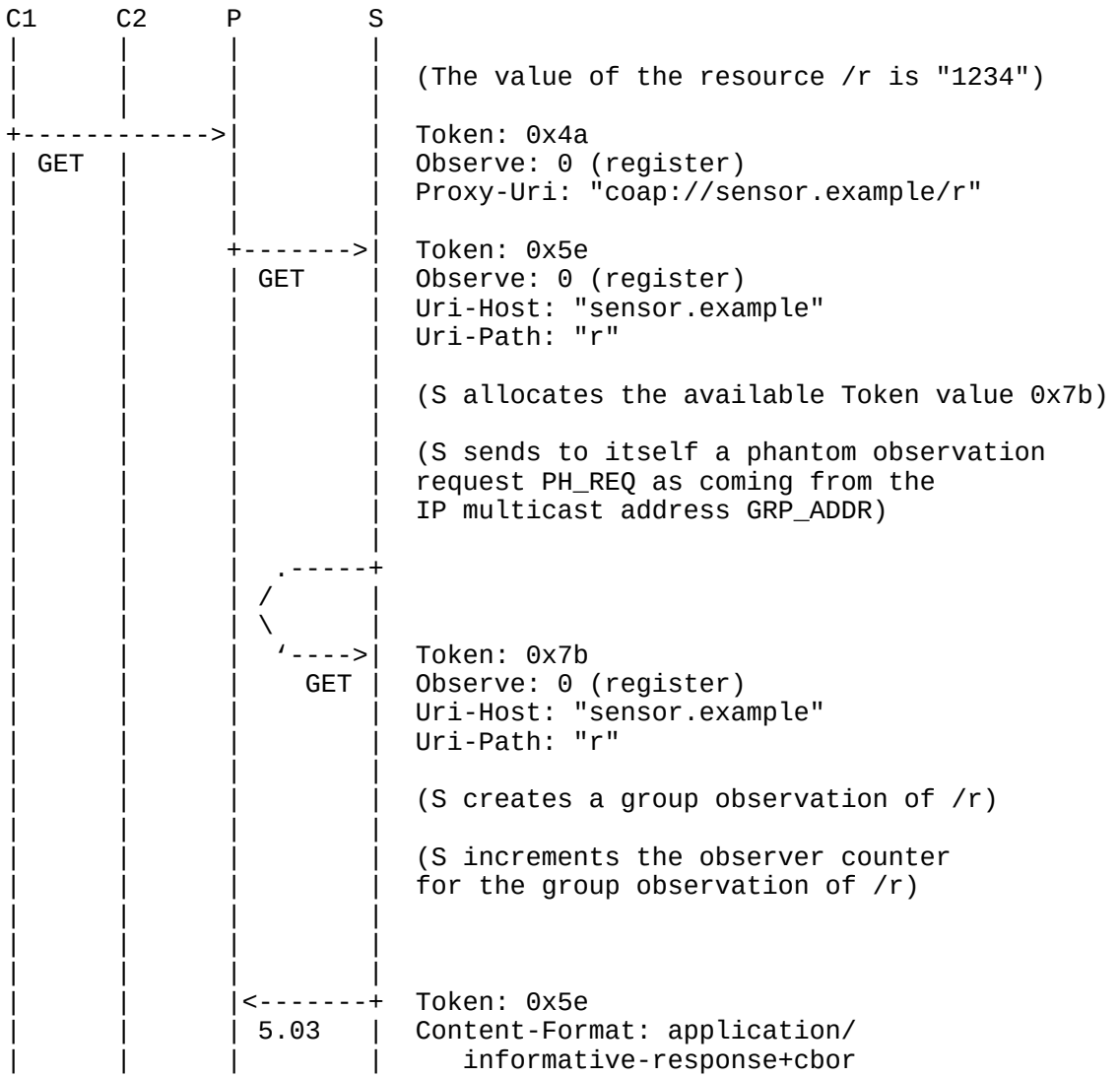
    When no interested origin clients remain, the proxy can simply forget about being part of the group observation for the target resource at the server, like an origin client would do (see Section 5.4 of [I-D.ietf-core-observe-multicast-notifications]).

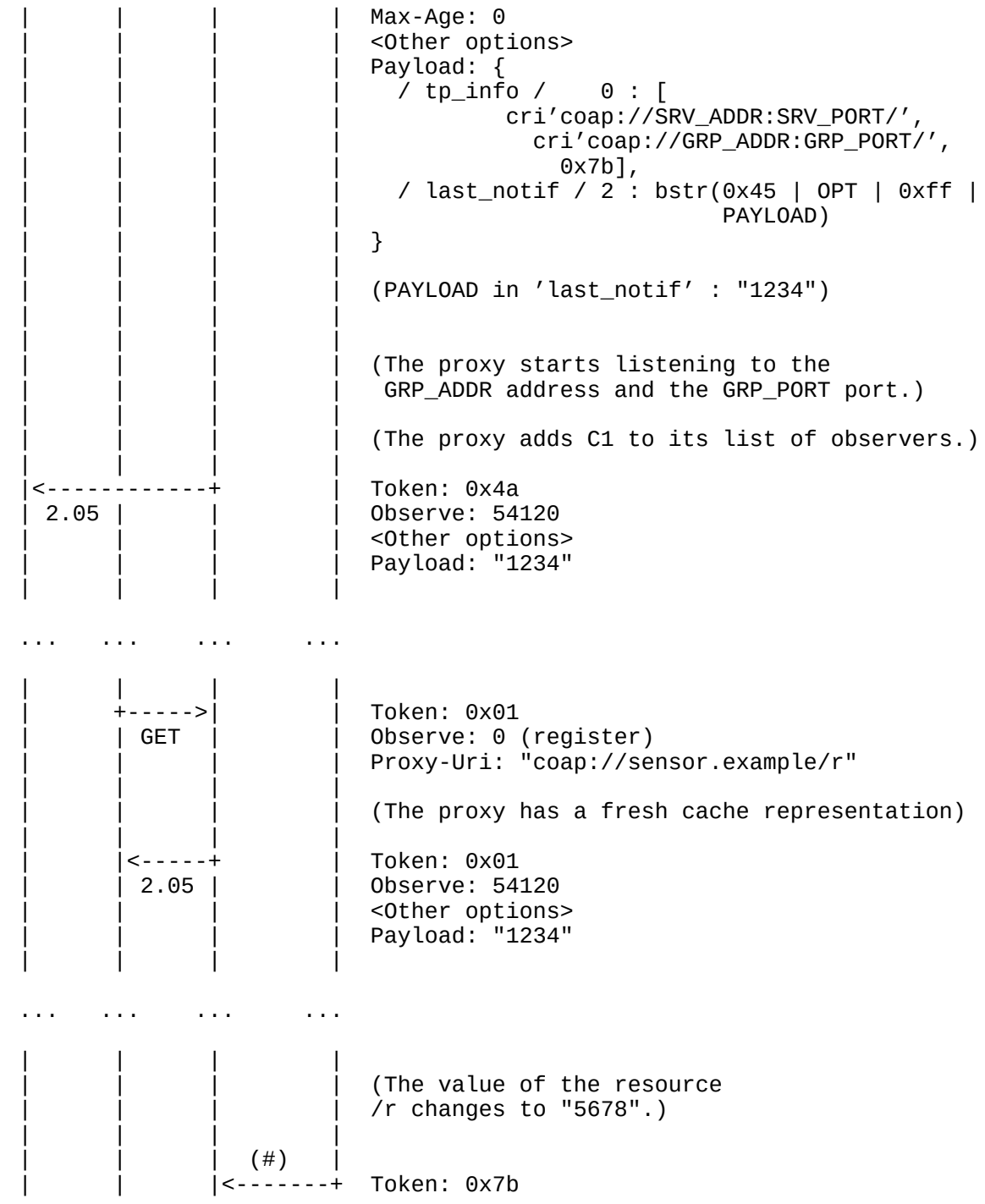
  - If the multicast notification is protected end-to-end by using Group OSCORE (see Section 9 of [I-D.ietf-core-observe-multicast-notifications] and Section 4 of this document), then the Feedback-Divider Option is protected end-to-end as well, and is thus hidden from the proxy. As a consequence, the proxy forwards the notification to (the previous hop towards) any of the origin clients, each of which answers to the server if it picks a random number I equal to 0.

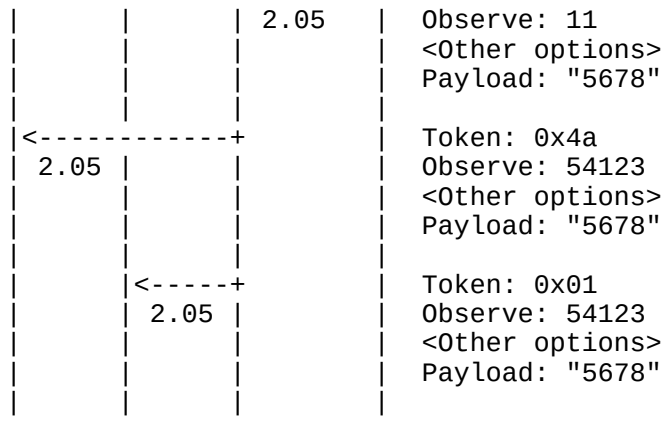
6. Example with a Proxy

This section provides an example where a proxy P is used between the clients and the server. The same assumptions and notation used in Section 7 of [I-D.ietf-core-observe-multicast-notifications] are used for this example. In addition, the proxy has address PRX_ADDR and listens to the port number PRX_PORT.

Unless explicitly indicated, all messages transmitted on the wire are sent over unicast.




Max-Age: 0
<Other options>
Payload: {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b],
/ last_notif / 2 : bstr(0x45 | OPT | 0xff |
PAYLOAD)
}
(PAYLOAD in 'last_notif' : "1234")
(The proxy starts listening to the
GRP_ADDR address and the GRP_PORT port.)
(The proxy adds C1 to its list of observers.)
Token: 0x4a
2.05
Observe: 54120
<Other options>
Payload: "1234"
...
...
...
...
Token: 0x01
GET
Observe: 0 (register)
Proxy-Uri: "coap://sensor.example/r"
(The proxy has a fresh cache representation)
Token: 0x01
2.05
Observe: 54120
<Other options>
Payload: "1234"
...
...
...
...
(The value of the resource
/r changes to "5678".)
(#)
Token: 0x7b



(#) Sent over IP multicast to GROUP_ADDR:GROUP_PORT.

Figure 1: Example of Group Observation with a Proxy

Note that the proxy has all the information to understand the observation request from C2 and can immediately start to serve the still fresh values.

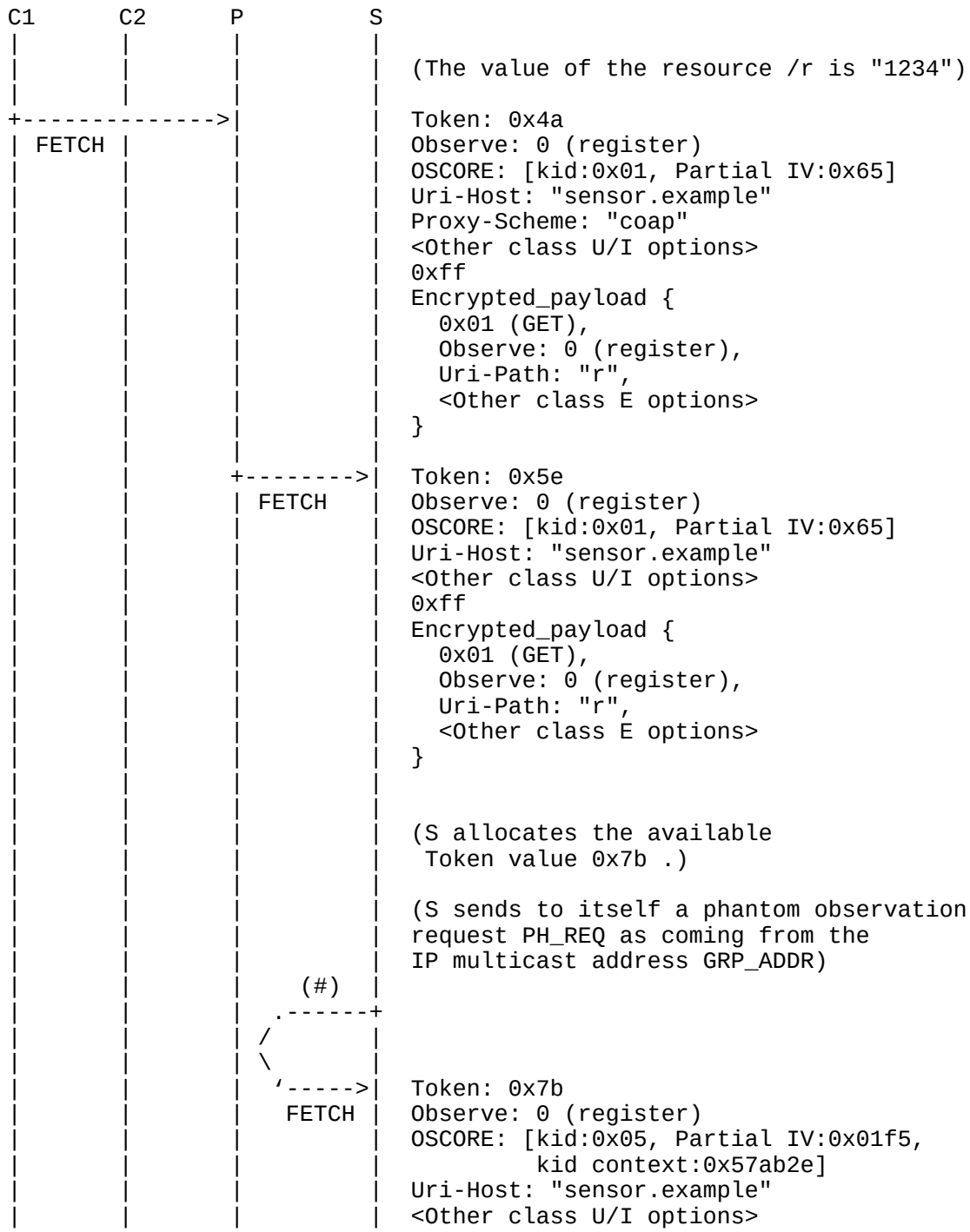
This behavior is mandated by Section 5 of [RFC7641], i.e., the proxy registers itself only once with the next hop and fans out the notifications that it receives to all the registered clients.

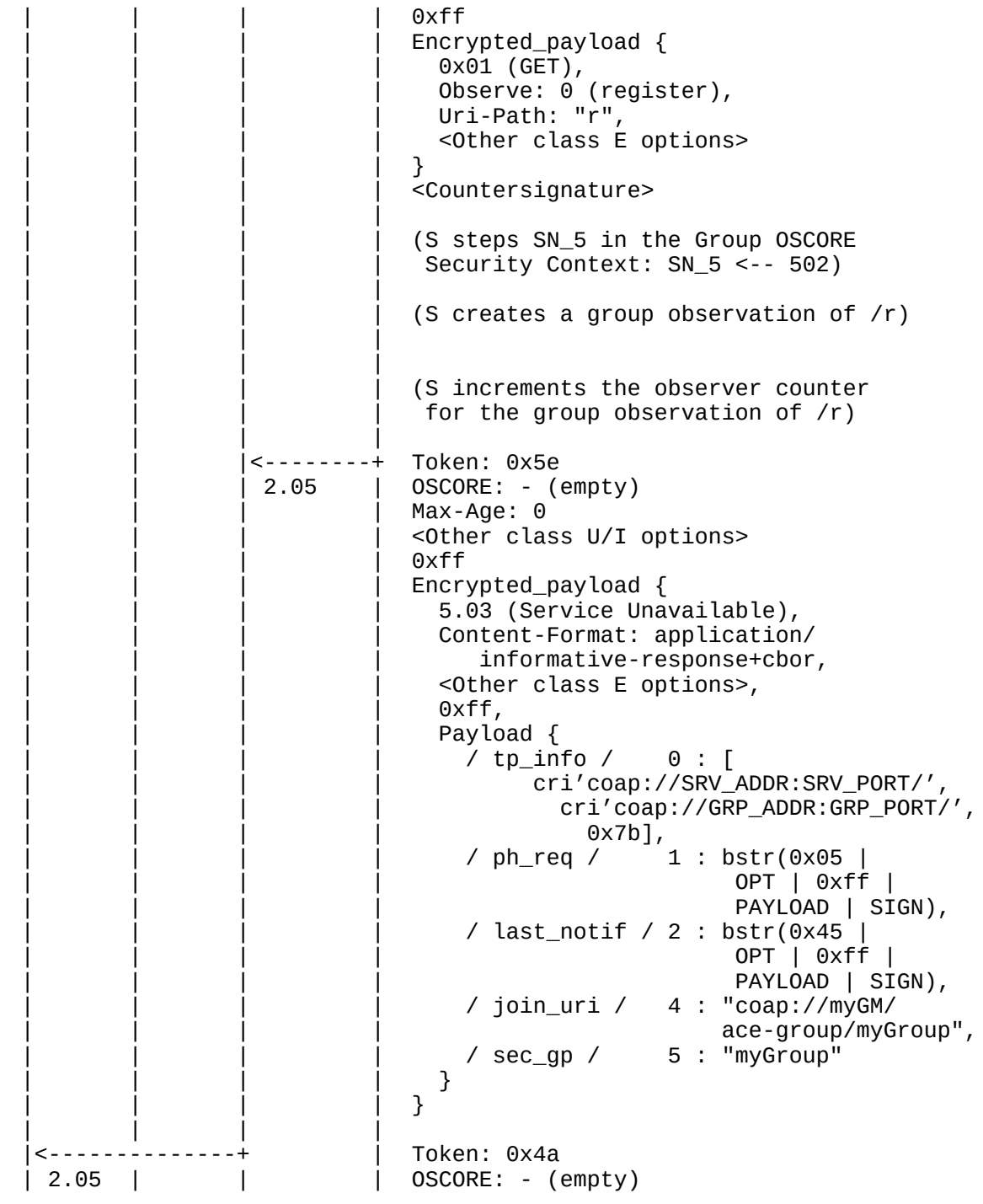
7. Example with a Proxy and with Group OSCORE

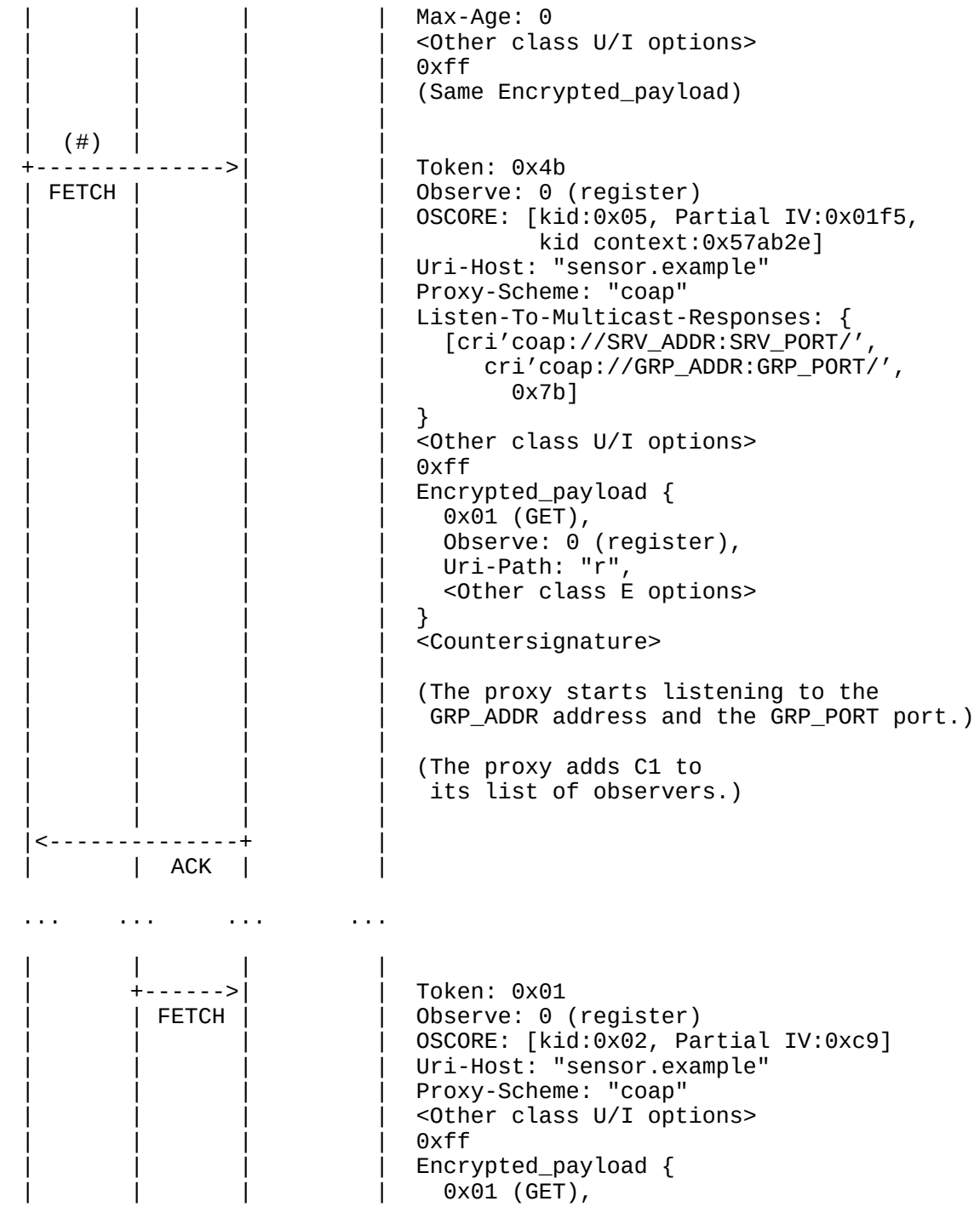
This section provides an example where a proxy P is used between the clients and the server, and Group OSCORE is used to protect multicast notifications end-to-end between the server and the clients.

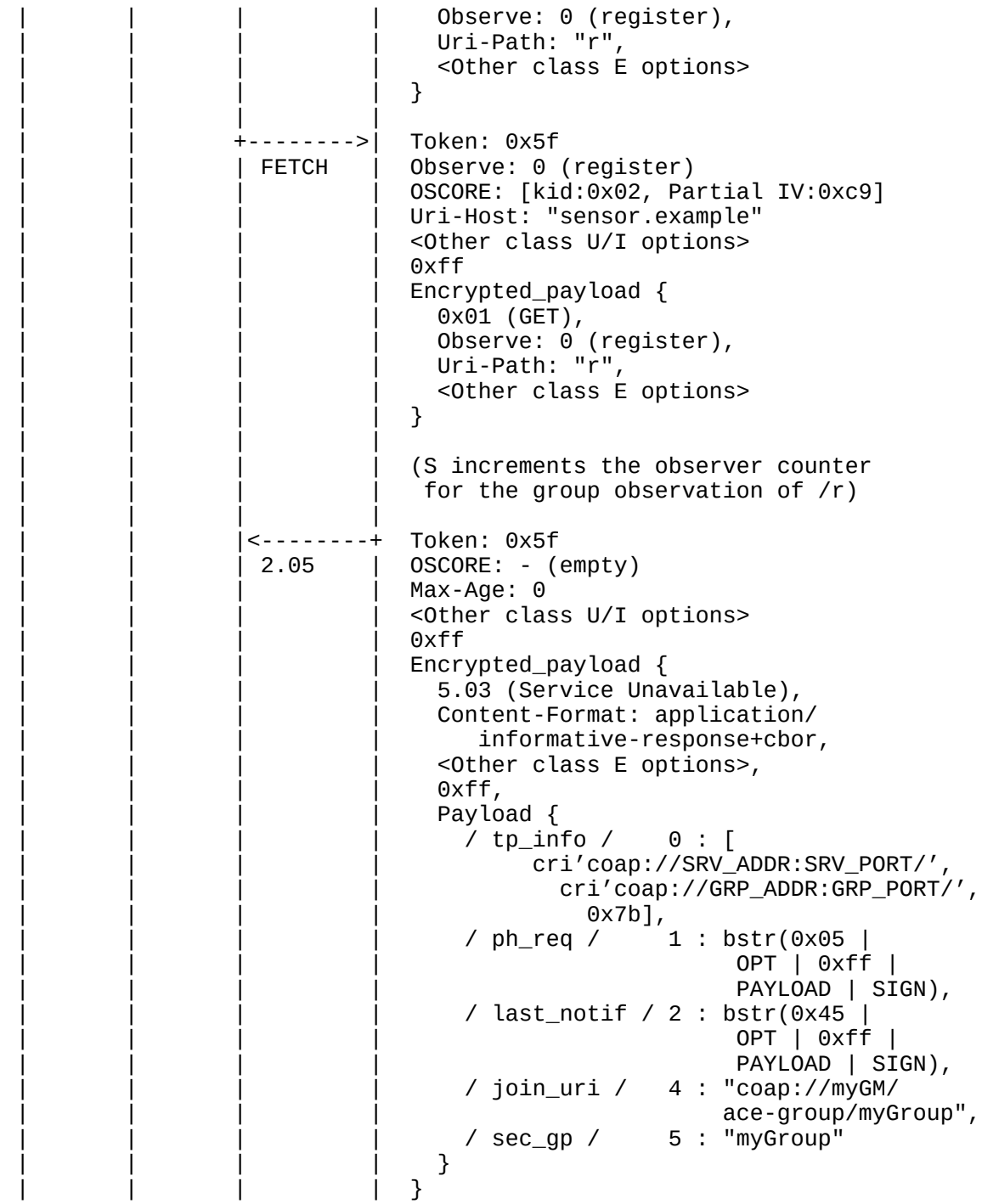
The same assumptions and notation used in Section 10 of [I-D.ietf-core-observe-multicast-notifications] are used for this example. In addition, the proxy has address PRX_ADDR and listens to the port number PRX_PORT.

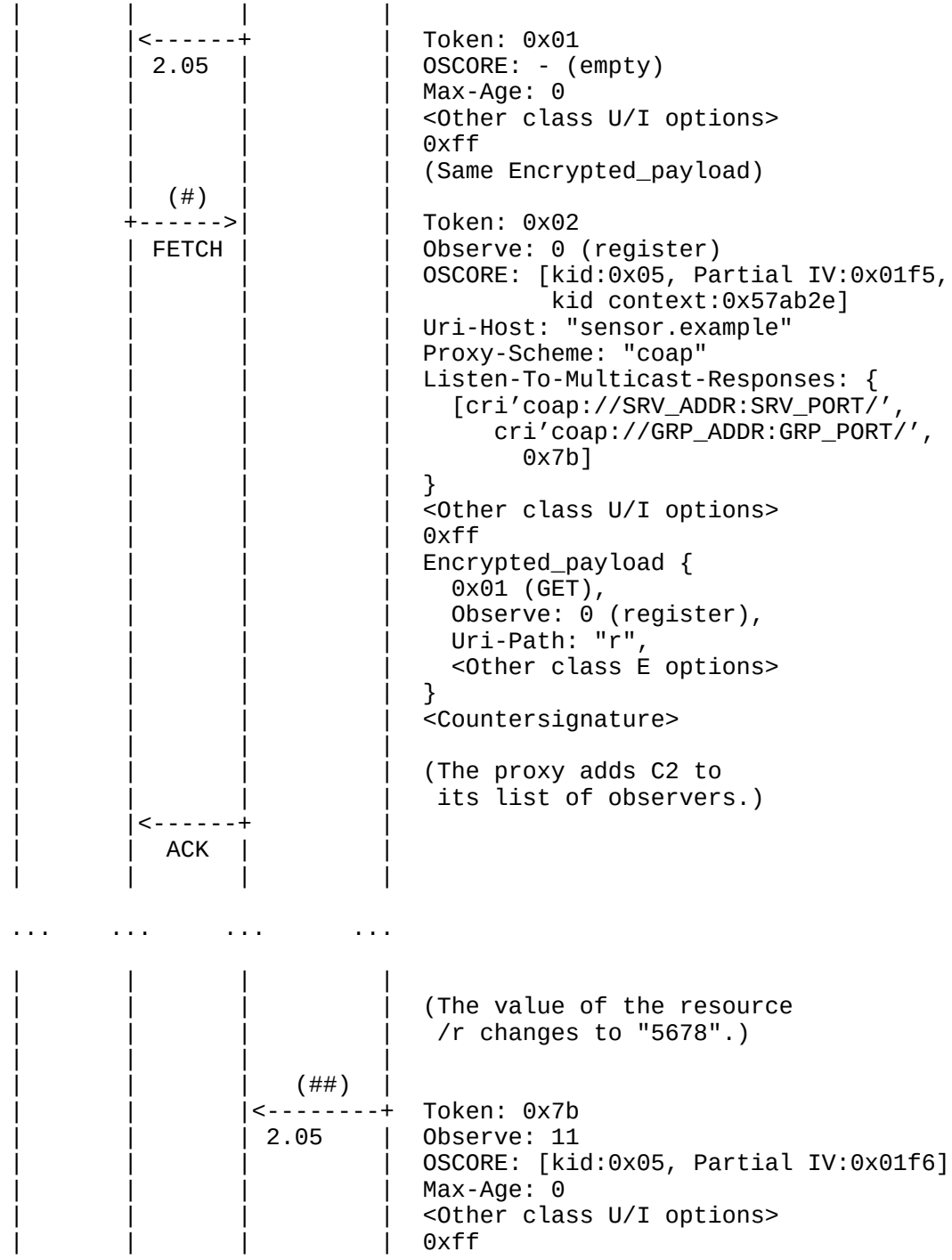
Unless explicitly indicated, all messages transmitted on the wire are sent over unicast and protected with OSCORE end-to-end between a client and the server.

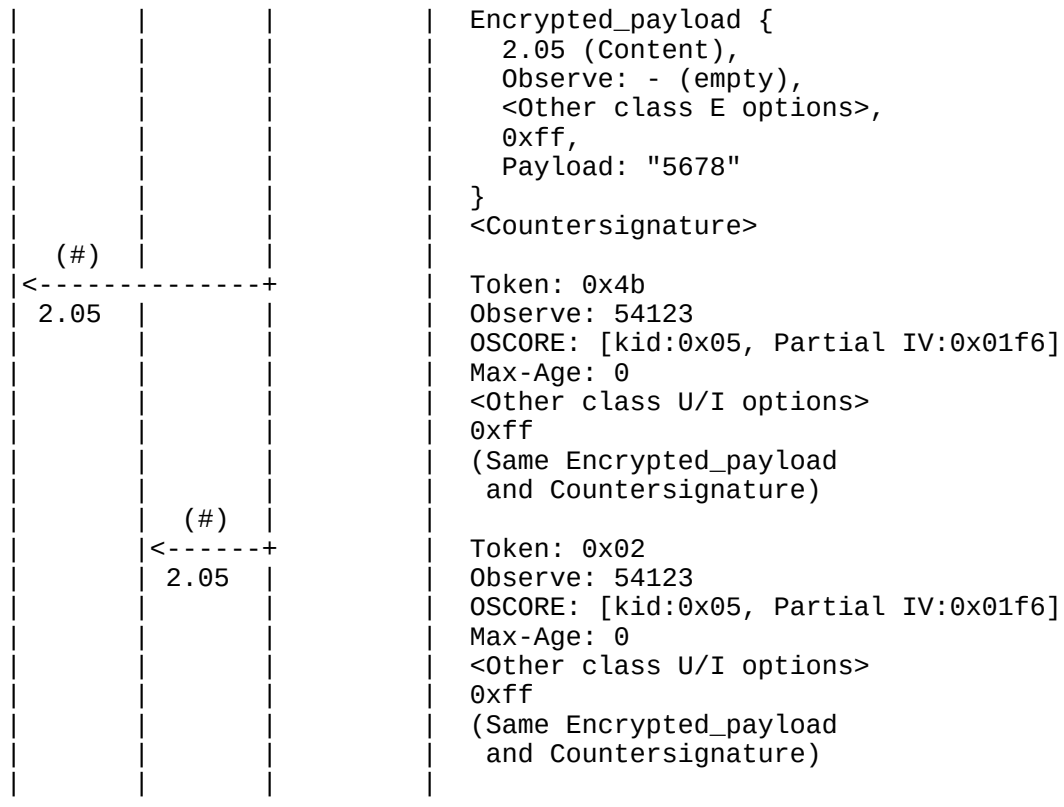

C1
C2
P
S
(The value of the resource /r is "1234")
Token: 0x4a
FETCH
Observe: 0 (register)
OSCORE: [kid:0x01, Partial IV:0x65]
Uri-Host: "sensor.example"
Proxy-Scheme: "coap"
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
Token: 0x5e
FETCH
Observe: 0 (register)
OSCORE: [kid:0x01, Partial IV:0x65]
Uri-Host: "sensor.example"
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
(S allocates the available
Token value 0x7b .)
(S sends to itself a phantom observation
request PH_REQ as coming from the
IP multicast address GRP_ADDR)
(#)
Token: 0x7b
FETCH
Observe: 0 (register)
OSCORE: [kid:0x05, Partial IV:0x01f5,
kid context:0x57ab2e]
Uri-Host: "sensor.example"
<Other class U/I options>


0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
<Countersignature>
(S steps SN_5 in the Group OSCORE
Security Context: SN_5 <-- 502)
(S creates a group observation of /r)
(S increments the observer counter
for the group observation of /r)
Token: 0x5e
2.05
OSCORE: - (empty)
Max-Age: 0
<Other class U/I options>
0xff
Encrypted_payload {
5.03 (Service Unavailable),
Content-Format: application/
informative-response+cbor,
<Other class E options>,
0xff,
Payload {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b],
/ ph_req / 1 : bstr(0x05 |
OPT | 0xff |
PAYLOAD | SIGN),
/ last_notif / 2 : bstr(0x45 |
OPT | 0xff |
PAYLOAD | SIGN),
/ join_uri / 4 : "coap://myGM/
ace-group/myGroup",
/ sec_gp / 5 : "myGroup"
}
}
Token: 0x4a
2.05
OSCORE: - (empty)


Max-Age: 0
<Other class U/I options>
0xff
(Same Encrypted_payload)
(#)
FETCH
Token: 0x4b
Observe: 0 (register)
OSCORE: [kid:0x05, Partial IV:0x01f5,
kid context:0x57ab2e]
Uri-Host: "sensor.example"
Proxy-Scheme: "coap"
Listen-To-Multicast-Responses: {
[cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b]
}
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
<Countersignature>
(The proxy starts listening to the
GRP_ADDR address and the GRP_PORT port.)
(The proxy adds C1 to
its list of observers.)
ACK
Token: 0x01
Observe: 0 (register)
OSCORE: [kid:0x02, Partial IV:0xc9]
Uri-Host: "sensor.example"
Proxy-Scheme: "coap"
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),


Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
Token: 0x5f
FETCH
Observe: 0 (register)
OSCORE: [kid:0x02, Partial IV:0xc9]
Uri-Host: "sensor.example"
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
(S increments the observer counter
for the group observation of /r)
Token: 0x5f
2.05
OSCORE: - (empty)
Max-Age: 0
<Other class U/I options>
0xff
Encrypted_payload {
5.03 (Service Unavailable),
Content-Format: application/
informative-response+cbor,
<Other class E options>,
0xff,
Payload {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b],
/ ph_req / 1 : bstr(0x05 |
OPT | 0xff |
PAYLOAD | SIGN),
/ last_notif / 2 : bstr(0x45 |
OPT | 0xff |
PAYLOAD | SIGN),
/ join_uri / 4 : "coap://myGM/
ace-group/myGroup",
/ sec_gp / 5 : "myGroup"
}
}


2.05
(#)
FETCH
Token: 0x01
OSCORE: - (empty)
Max-Age: 0
<Other class U/I options>
0xff
(Same Encrypted_payload)
Token: 0x02
Observe: 0 (register)
OSCORE: [kid:0x05, Partial IV:0x01f5,
kid context:0x57ab2e]
Uri-Host: "sensor.example"
Proxy-Scheme: "coap"
Listen-To-Multicast-Responses: {
[cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b]
}
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
<Countersignature>
(The proxy adds C2 to
its list of observers.)
ACK
(The value of the resource
/r changes to "5678".)
(##)
2.05
Token: 0x7b
Observe: 11
OSCORE: [kid:0x05, Partial IV:0x01f6]
Max-Age: 0
<Other class U/I options>
0xff



(#) Sent over unicast, and protected with Group OSCORE end-to-end between the server and the clients.

(##) Sent over IP multicast to GROUP_ADDR:GROUP_PORT, protected end-to-end with Group OSCORE between the server and the clients.

Figure 2: Example of Group Observation with a Proxy and Group OSCORE

Unlike in the unprotected example in Section 6, the proxy does not have all the information to perform request deduplication and can only recognize the identical request once the client sends the ticket request.

8. Example with a Proxy and with Deterministic Requests

This section provides an example where a proxy P is used between the clients and the server, and Group OSCORE is used to protect multicast notifications end-to-end between the server and the clients.

In addition, the phantom request is especially a Deterministic Request (see Appendix D of [I-D.ietf-core-observe-multicast-notifications]), which is protected with the pairwise mode of Group OSCORE as defined in [I-D.ietf-core-cacheable-oscore].

Since the server replies to such a Deterministic Request with an informative response that is not protected (see Appendix D of [I-D.ietf-core-observe-multicast-notifications]), the proxy is able to retrieve from the informative response everything needed to set itself as an observer in the group observation and to start listening to multicast notifications.

In particular, each client sends the Deterministic Request to the proxy as a ticket request (see Section 4). However, differently from what is defined in Section 4 where the ticket request is not a Deterministic Request, the clients do not include a Listen-to-Multicast-Responses Option. This results in the proxy forwarding the ticket request (i.e., the phantom observation request) to the server and obtaining the information required to listen to multicast notifications, unless the proxy has already set itself to do so. Also, the proxy will be able to serve multicast notifications from its cache as per [I-D.ietf-core-cacheable-oscore].

Appendix D of [I-D.ietf-core-observe-multicast-notifications] discusses how, when using a Deterministic Request as a phantom observation request, the observer counter at the server (see Section 4 of [I-D.ietf-core-observe-multicast-notifications]) is not reliably incremented when new clients start participating in the group observation. The same applies also if a proxy is deployed.

That is, the origin server increments its observer counter after having sent the informative response to the proxy, as a reply to the Deterministic Request forwarded to the origin server on behalf of the first origin client that contacted the proxy. After that, the same Deterministic Request sent by any origin client will not be forwarded to the origin server, but will instead produce a cache hit at the proxy that will serve the client accordingly. Hence, the observer counter at the server is not further incremented as additional, new origin clients start participating in the group observation through the proxy.

Also in this case, the security identity associated with the sender of any Deterministic Request in the OSCORE group is exactly the same one, i.e., the pair (SID, OSCORE ID Context), where SID is the OSCORE Sender ID of the Deterministic Client in the OSCORE group, which all the clients in the group rely on to produce Deterministic Requests.

8.1. Assumptions and Walkthrough

The example provided in this appendix as reflected by the message exchange shown in Section 8.2 assumes the following.

1. The OSCORE group supports Deterministic Requests. Thus, the server creates the phantom request as a Deterministic Request [I-D.ietf-core-cacheable-oscore], stores it locally as one of its issued phantom requests, and starts the corresponding group observation.

2. The server makes the phantom request available through other means (e.g., a pub-sub broker), together with the transport-specific information for listening to multicast notifications bound to the phantom request (see Appendix A of [I-D.ietf-core-observe-multicast-notifications]).

3. Since the phantom request is a Deterministic Request, the server can more efficiently make it available in its smaller, plain version. The clients can obtain it from the particular alternative source and protect it as per Section 3 of [I-D.ietf-core-cacheable-oscore], thus all computing the same Deterministic Request to be used as phantom observation request.

4. If a client does not rely on a proxy between itself and the server, it simply sets the group observation and starts listening to multicast notifications. Building on Step 2 above, the same would happen if the phantom request was not specifically a Deterministic Request.

5. If a client relies on a proxy between itself and the server, it uses the phantom request as a ticket request (see Section 4). However, unlike for the case in Section 4 where the ticket request is not a Deterministic Request, the client does not include a Listen-to-Multicast-Responses Option in the phantom request sent to the proxy.

6. Unlike for the case in Section 4, the proxy does not know that the request is exactly a ticket request for subscribing to multicast notifications. Thus, the proxy simply forwards the ticket request to the server like it normally would.

7. The server receives the ticket request, which is a deviation from the case where the ticket request is not a Deterministic Request and stops at the proxy (see Section 4). Then, the server recognizes the phantom request among the stored ones, through a byte-by-byte comparison of the incoming message minus the transport-related fields (see Appendix D of [I-D.ietf-core-observe-multicast-notifications]). Consequently, the server does not perform any Group OSCORE processing on it.

8. The server replies with an unprotected informative response (see Section 4.2 of [I-D.ietf-core-observe-multicast-notifications]), including: the transport-specific information, (optionally) the phantom request, and (optionally) the latest notification.

   Note that the phantom request can be omitted, since it is the deterministic phantom request from the client, and thus "in terms of transport-independent information, identical to the registration request from the client" (see Section 4.2 of [I-D.ietf-core-observe-multicast-notifications]).

9. From the received informative response, the proxy retrieves everything needed to set itself as an observer in the group observation and it starts listening to multicast notifications. If the informative response includes a latest notification, the proxy caches it and forwards it back to the client. Otherwise, the proxy replies with an empty ACK (if it has not done it already and the request from the client was a Confirmable message).

10. Like for the case with a non-deterministic phantom request in Section 4, the proxy fans out the multicast notifications to the origin clients as they come. Also, as new clients following the first one contact the proxy, the latter does not have to contact the server again as in Section 4, since the deterministic phantom request would produce a cache hit as per [I-D.ietf-core-cacheable-oscore]. Thus, the proxy can serve such clients with the latest fresh multicast notification from its cache.

8.2. Message Exchange

The same assumptions and notation used in Section 10 of [I-D.ietf-core-observe-multicast-notifications] are used for this example. As a recap of some specific values:

* Two clients C1 and C2 register to observe a resource /r at a server S, which has address SRV_ADDR and listens to the port number SRV_PORT. Before the following exchanges occur, no clients are observing the resource /r , which has value "1234".

* The server S sends multicast notifications to the IP multicast address GRP_ADDR and port number GRP_PORT, and starts the group observation already after creating the deterministic phantom request to early disseminate.

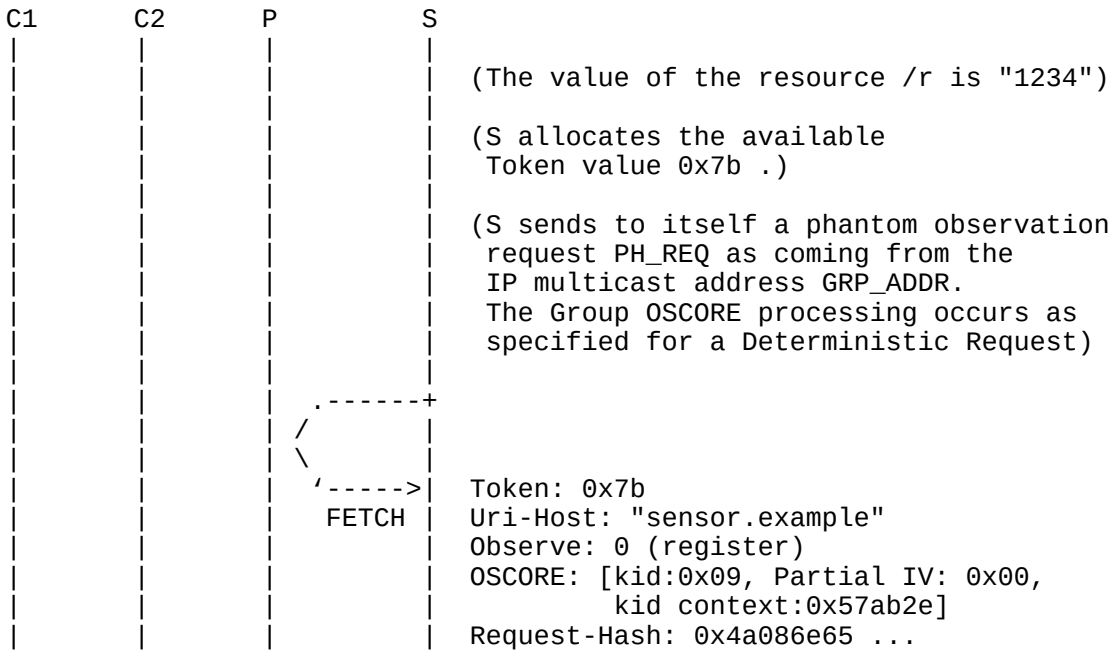
* S is a member of the OSCORE group with 'kid context' = 0x57ab2e as Group ID. In the OSCORE group, S has 'kid' = 0x05 as Sender ID and SN_5 = 501 (i.e., 0x01f5) as Sender Sequence Number.

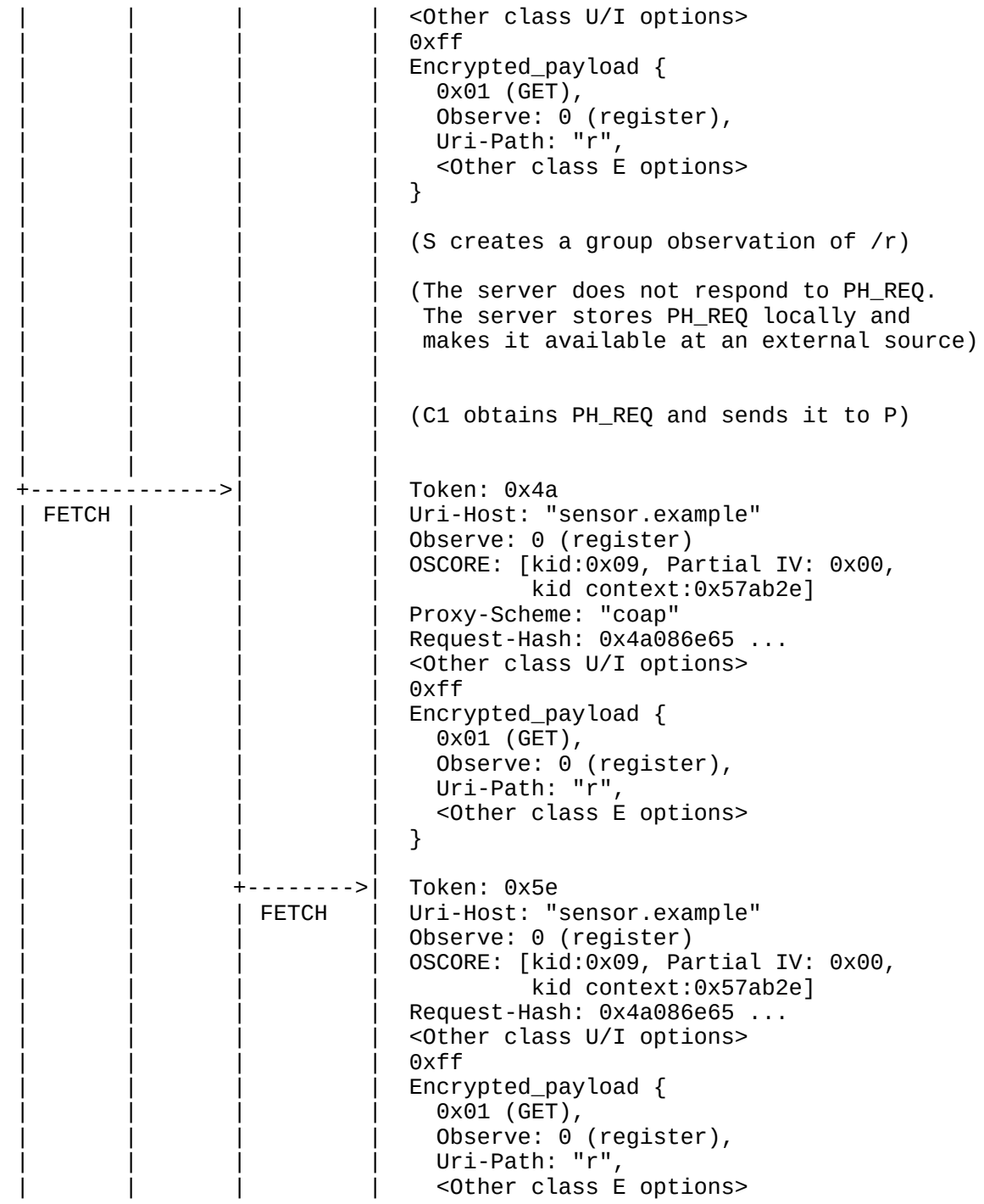
In addition:

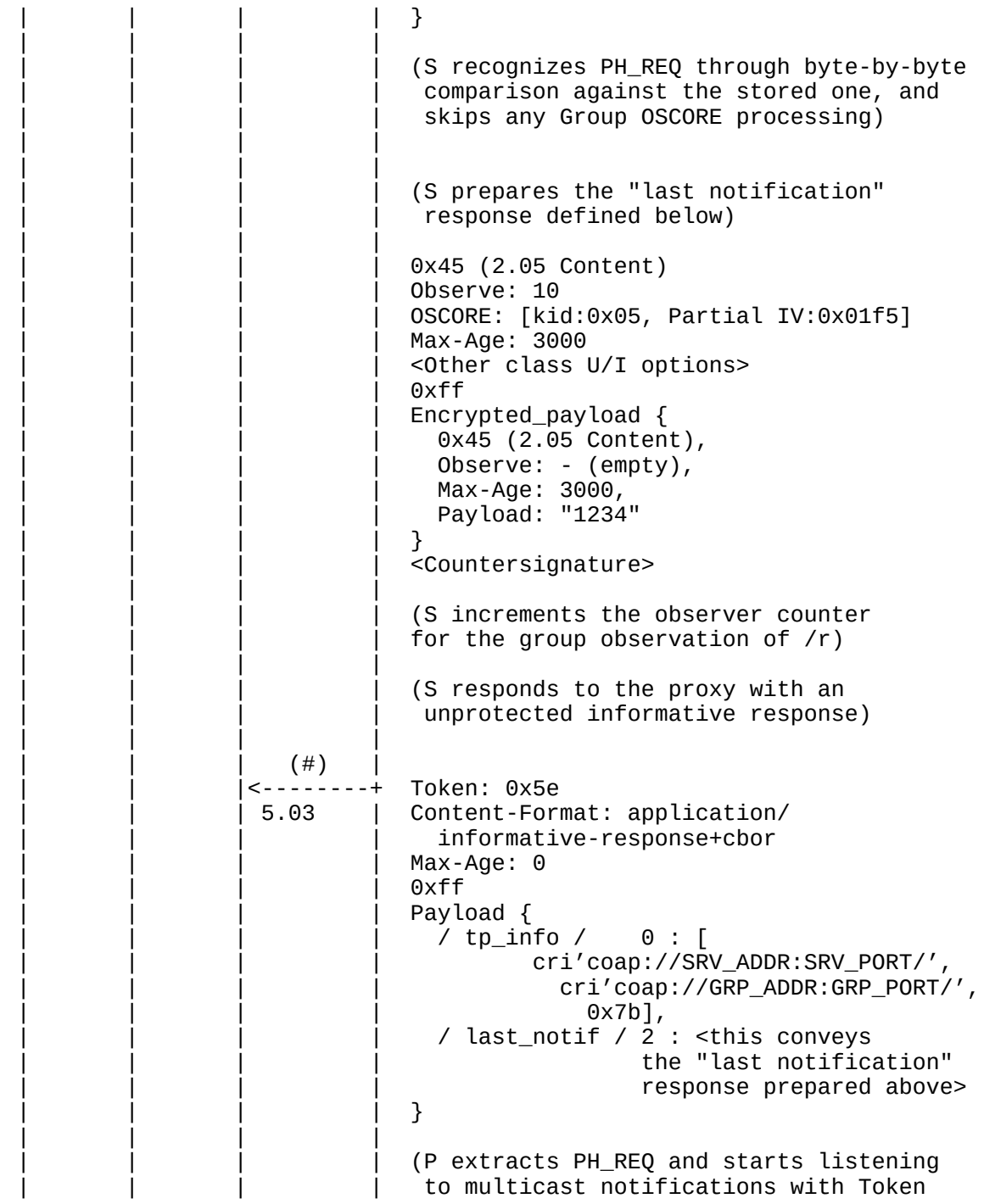
* The proxy has address PRX_ADDR and listens to the port number PRX_PORT.

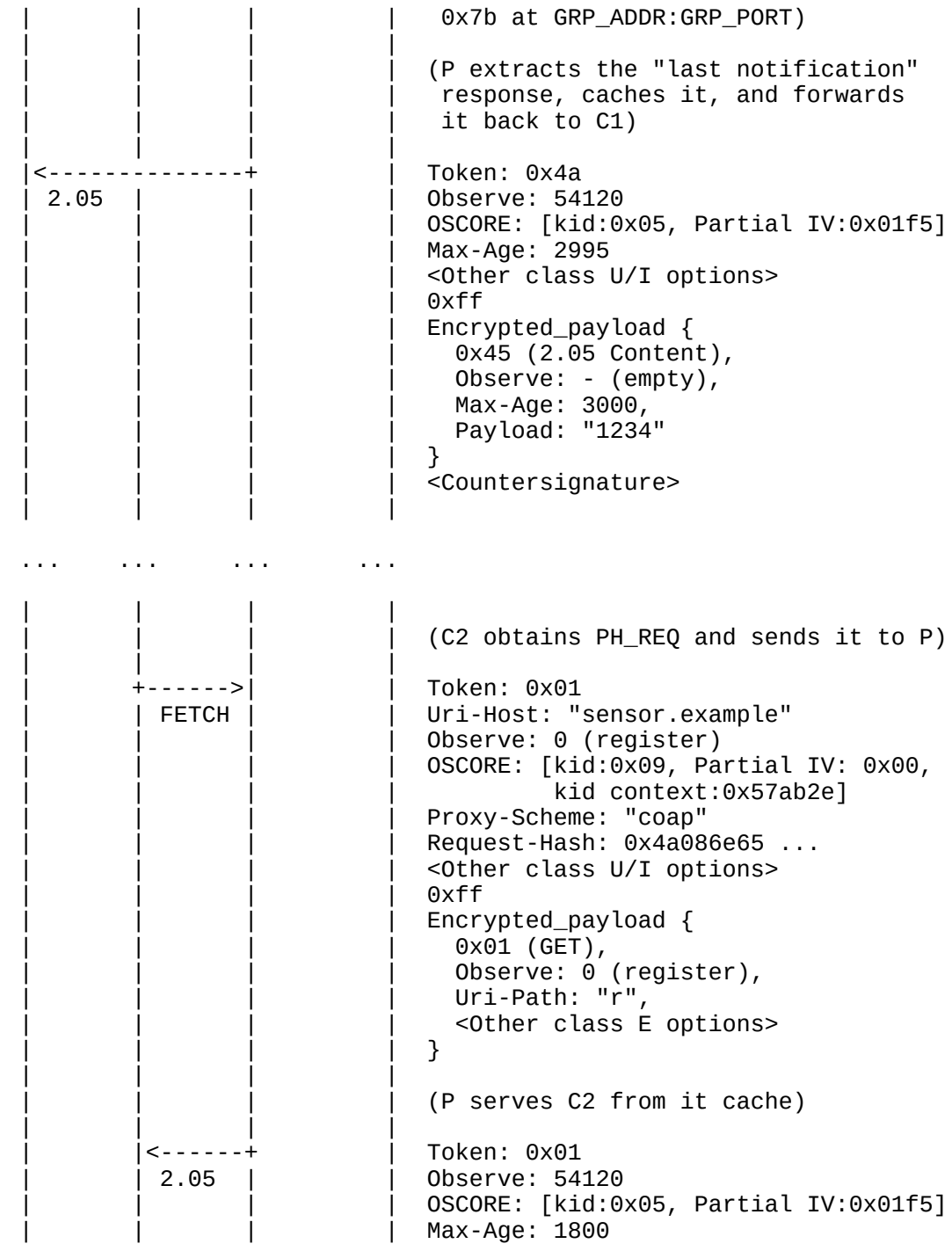
* The deterministic client in the OSCORE group has 'kid' = 0x09 as Sender ID.

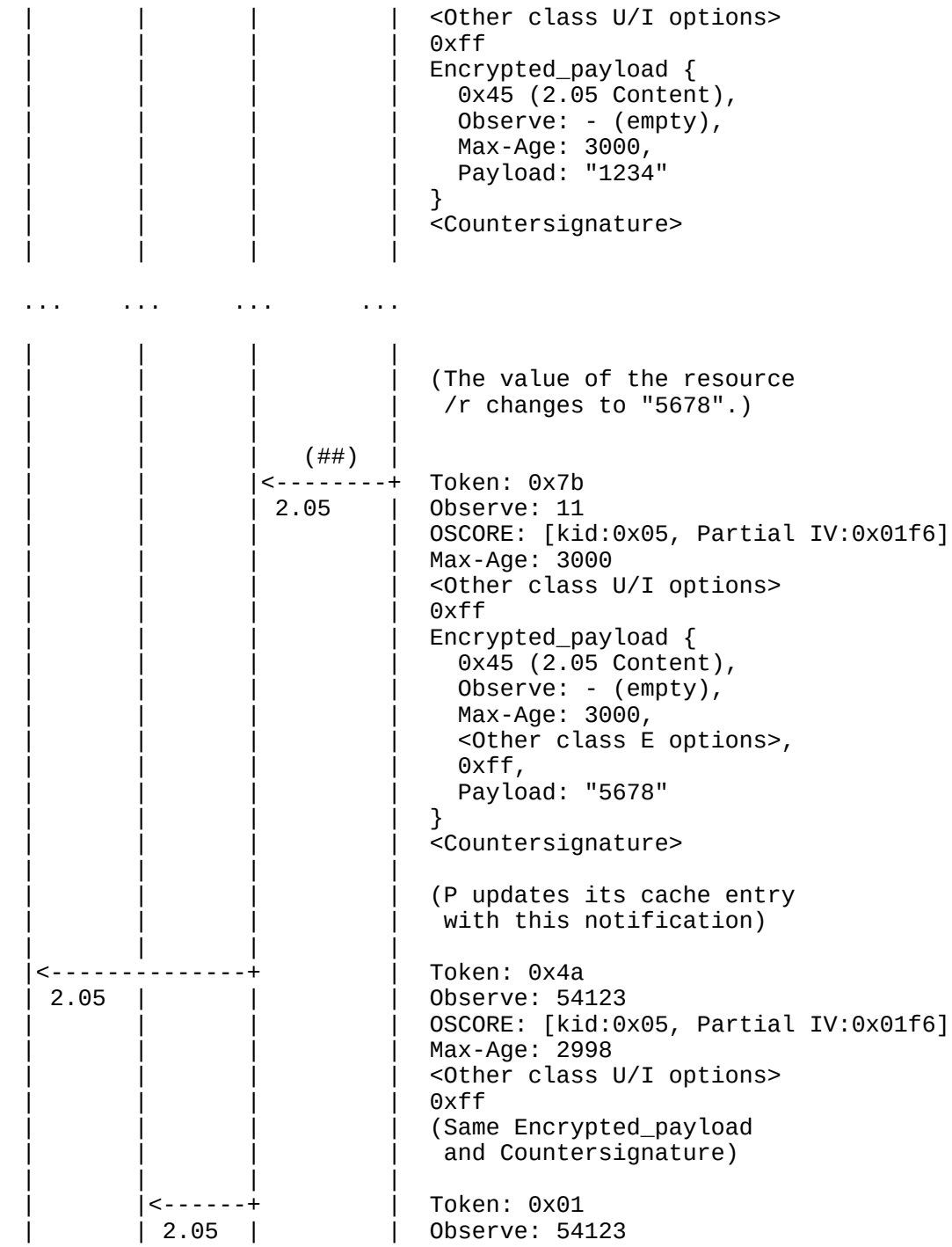
Unless explicitly indicated, all messages transmitted on the wire are sent over unicast and protected with Group OSCORE end-to-end between a client and the server.




<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
(S creates a group observation of /r)
(The server does not respond to PH_REQ.
The server stores PH_REQ locally and
makes it available at an external source)
(C1 obtains PH_REQ and sends it to P)
FETCH
Token: 0x4a
Uri-Host: "sensor.example"
Observe: 0 (register)
OSCORE: [kid:0x09, Partial IV: 0x00,
kid context:0x57ab2e]
Proxy-Scheme: "coap"
Request-Hash: 0x4a086e65 ...
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
FETCH
Token: 0x5e
Uri-Host: "sensor.example"
Observe: 0 (register)
OSCORE: [kid:0x09, Partial IV: 0x00,
kid context:0x57ab2e]
Request-Hash: 0x4a086e65 ...
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>


}
(S recognizes PH_REQ through byte-by-byte
comparison against the stored one, and
skips any Group OSCORE processing)
(S prepares the "last notification"
response defined below)
0x45 (2.05 Content)
Observe: 10
OSCORE: [kid:0x05, Partial IV:0x01f5]
Max-Age: 3000
<Other class U/I options>
0xff
Encrypted_payload {
0x45 (2.05 Content),
Observe: - (empty),
Max-Age: 3000,
Payload: "1234"
}
<Countersignature>
(S increments the observer counter
for the group observation of /r)
(S responds to the proxy with an
unprotected informative response)
(#)
5.03
Token: 0x5e
Content-Format: application/
informative-response+cbor
Max-Age: 0
0xff
Payload {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b],
/ last_notif / 2 : <this conveys
the "last notification"
response prepared above>
}
(P extracts PH_REQ and starts listening
to multicast notifications with Token


0x7b at GRP_ADDR:GRP_PORT)
(P extracts the "last notification"
response, caches it, and forwards
it back to C1)
2.05
Token: 0x4a
Observe: 54120
OSCORE: [kid:0x05, Partial IV:0x01f5]
Max-Age: 2995
<Other class U/I options>
0xff
Encrypted_payload {
0x45 (2.05 Content),
Observe: - (empty),
Max-Age: 3000,
Payload: "1234"
}
<Countersignature>
(C2 obtains PH_REQ and sends it to P)
FETCH
Token: 0x01
Uri-Host: "sensor.example"
Observe: 0 (register)
OSCORE: [kid:0x09, Partial IV: 0x00,
kid context:0x57ab2e]
Proxy-Scheme: "coap"
Request-Hash: 0x4a086e65 ...
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
(P serves C2 from it cache)
2.05
Token: 0x01
Observe: 54120
OSCORE: [kid:0x05, Partial IV:0x01f5]
Max-Age: 1800


<Other class U/I options>
0xff
Encrypted_payload {
0x45 (2.05 Content),
Observe: - (empty),
Max-Age: 3000,
Payload: "1234"
}
<Countersignature>
(The value of the resource
/r changes to "5678".)
(##)
2.05
Token: 0x7b
Observe: 11
OSCORE: [kid:0x05, Partial IV:0x01f6]
Max-Age: 3000
<Other class U/I options>
0xff
Encrypted_payload {
0x45 (2.05 Content),
Observe: - (empty),
Max-Age: 3000,
<Other class E options>,
0xff,
Payload: "5678"
}
<Countersignature>
(P updates its cache entry
with this notification)
2.05
Token: 0x4a
Observe: 54123
OSCORE: [kid:0x05, Partial IV:0x01f6]
Max-Age: 2998
<Other class U/I options>
0xff
(Same Encrypted_payload
and Countersignature)
2.05
Token: 0x01
Observe: 54123

```
 |       |       |         |  OSCORE: [kid:0x05, Partial IV:0x01f6]
 |       |       |         |  Max-Age: 2996
 |       |       |         |  <Other class U/I options>
 |       |       |         |  0xff
 |       |       |         |  (Same Encrypted_payload
 |       |       |         |   and Countersignature)
 |       |       |         |
```

(#) Sent over unicast and unprotected.

(##) Sent over IP multicast to GROUP_ADDR:GROUP_PORT, protected end-to-end with Group OSCORE between the server and the clients.

Figure 3: Example of Group Observation with a Proxy and Group OSCORE, where the Phantom Request is a Deterministic Request

9. Example with a Reverse-Proxy and with Deterministic Requests

This section describes an example where specifically a reverse-proxy PRX is used between the clients and the server (see Section 5.7.3 of [RFC7252]).

Like for the example in Section 8, the phantom request is especially a Deterministic Request (see Appendix D of [I-D.ietf-core-observe-multicast-notifications]), which is protected with the pairwise mode of Group OSCORE as defined in [I-D.ietf-core-cacheable-oscore].

The same assumptions compiled in Section 8.1 apply in this scenario too, with the following differences:

* Assumption (2): when the server makes the phantom request available through other means (see Appendix A of [I-D.ietf-core-observe-multicast-notifications]), the accompanying group observation data does _not_ specify client-side, transport-specific information for listening to multicast notifications bound to the phantom request.

* Assumption (4): this assumption does not apply, since all the clients rely on PRX, although they are not aware to communicate with a proxy.

Furthermore, the following assumptions apply to this scenario:

* The proxy has address PRX_ADDR and listens to the port number PRX_PORT. In particular, PRX exposes PRX_ADDR and PRX_PORT to clients when acting as stand-in for the server.

That is, a request sent with destination address PRX_ADDR and port number PRX_PORT will reach PRX, which forwards the request to the server.

* The server knows the address PRX_ADDR and port number PRX_PORT that PRX exposes to clients.

* When the server makes the phantom request available through other means (see Appendix A of [I-D.ietf-core-observe-multicast-notifications]), the accompanying group observation data is such that:

  - It provides server-side, transport-specific information, which consists of the address PRX_ADDR and port number PRX_PORT associated with PRX.

  - It does not provide any client-side, transport-specific information.

  Assuming that the group information data has a format consistent with the 'tp_info' array of the informative response (see Section 4.2.1 of [I-D.ietf-core-observe-multicast-notifications]), this means that the 'tp_info' array includes only the 'tpi_server' element specifying a CRI with addressing information PRX_ADDR and PRX_PORT (i.e., targeting PRX). That is, 'tp_info' does not include the 'tpi_details' element, regardless of what is expected as per the transport used.

9.1. Taking Part in Group Observations

The rest of this section describes how a client can take part in a group observation.

If any of the following conditions does not hold, then the client first performs the initialization procedure described in Section 9.1.1.

* The client has already obtained the group observation data specifying the deterministic phantom request, which the server has made available through other means (see Appendix A of [I-D.ietf-core-observe-multicast-notifications]).

* The client is already a member of the correct OSCORE group.

The main process consists of the following steps.

1. From the group observation data, the client knows the deterministic phantom request PH_REQ, the address PRX_ADDR, and the port number PRX_PORT, but no client-side, transport-specific information.

   In such a particular situation, the client sends PH_REQ with destination address PRX_ADDR and port number PRX_PORT, i.e., to PRX.

2. Upon receiving PH_REQ, PRX performs the same actions that are performed by the proxy in the scenario of Section 8.

   That is, if PH_REQ results in a cache hit at PRX, then PRX replies to the client with the latest multicast notification for the target resource from its cache and takes no further actions.

   Otherwise, PRX forwards PH_REQ to the server. After recognizing PH_REQ byte-by-byte, the server replies to PRX with an unprotected informative response, where 'tp_info' also includes the 'tpi_details' element, specifying the information to receive multicast notifications for the target resource. Based on such information, PRX starts listening to multicast notifications. If the informative response includes a latest notification, then PRX caches that notification and forwards it to the client.

Editor's note: add a figure showing an example of message exchange.

9.1.1. Client Initialization Procedure

The following early initialization procedure is performed by a client that does not have the group observation data and/or is not a member of the correct OSCORE group, before starting the main process described in Section 9.1.

The client is minimally provided with the pair (PRX_ADDR, PRX_PORT) associated with PRX, which the client believes to be associated with the origin server.

a. The client sends a traditional Observe registration request with destination address PRX_ADDR and port number PRX_PORT, i.e., to PRX. The request is protected with (Group) OSCORE, i.e., end-to-end between the client and the server.

b. PRX receives the request and forwards it to the server, as usual.

c. The server replies with a 5.03 (Service Unavailable) informative response. The response is protected with (Group) OSCORE, i.e., end-to-end between the client and the server. The payload of the response specifies the following parameters:

* The 'tp_info' parameter, within which the 'tpi_server' element is a CRI with addressing information PRX_ADDR and PRX_PORT (i.e., targeting PRX). The 'tp_info' parameter does not include the 'tpi_details' element, regardless of what is expected as per the transport used.

* The 'ph_req' parameter, conveying the deterministic phantom request PH_REQ.

* Optionally, parameters conveying information that the client can use for joining the OSCORE group if that has not happened yet (see Section 9.1 of [I-D.ietf-core-observe-multicast-notifications]), as well as the keying material used in the OSCORE group if the server is managing it (see Appendix C of [I-D.ietf-core-observe-multicast-notifications]).

d. PRX receives the protected informative response and forwards it to the client, as usual.

e. Upon receiving the protected informative response, the client takes its payload as the group observation data for the group observation of interest.

Per the instructions specified in the response, the client takes the necessary steps to join the correct OSCORE group, if it is not already a member.

10. Security Considerations

In addition to the security considerations from [I-D.ietf-core-observe-multicast-notifications], the following considerations hold for this document.

10.1. Listen-To-Multicast-Responses Option

The CoAP option Listen-To-Multicast-Responses defined in Section 4.1 is of class U for OSCORE and Group OSCORE [RFC8613][I-D.ietf-core-oscore-groupcomm].

This allows the proxy adjacent to the origin server to access the option value conveyed in a ticket request (see Section 4.2) and to retrieve from it the transport-specific information about a phantom request. By doing so, the proxy becomes able to configure an observation of the target resource and to receive multicast notifications that match the phantom request.

Any proxy in the chain, as well as further possible intermediaries or on-path active adversaries, are thus able to remove the option or alter its content, before the ticket request reaches the proxy adjacent to the origin server.

Removing the option would result in the proxy adjacent to the origin server to not configure the group observation, if that has not happened yet. In such a case, the proxy would not receive the corresponding multicast notifications to be forwarded back to the clients.

Altering the option content would result in the proxy adjacent to the origin server incorrectly configuring a group observation (e.g., as based on a wrong multicast IP address), hence preventing the correct reception of multicast notifications and their forwarding to the clients. Alternatively, it would result in the proxy configuring bogus group observations that are currently not active on the origin server.

In order to prevent what is described above, the ticket requests conveying the Listen-To-Multicast-Responses Option can be additionally protected hop-by-hop, e.g., by using OSCORE (see [I-D.ietf-core-oscore-capable-proxies]) and/or DTLS [RFC9147].

11. IANA Considerations

This document has the following actions for IANA.

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

11.1. CoAP Option Numbers Registry

IANA is asked to enter the following option number to the "CoAP Option Numbers" registry [CoAP.Option.Numbers] within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

Number	Name	Reference
TBD47	Listen-To-Multicast-Responses	[RFC-XXXX]

Table 2: Registrations in the CoAP Option Numbers Registry

For the Listen-To-Multicast-Responses Option, the preferred value range is 0-255. In particular, 47 is the preferred option number.

Note to RFC Editor: In the table above, please replace TBD47 with the registered option number. Then, please delete this paragraph and the previous paragraph.

12. References

12.1. Normative References

[CoAP.Option.Numbers]
IANA, "CoAP Option Numbers", <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#option-numbers>.

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

[I-D.ietf-core-observe-multicast-notifications]
Tiloca, M., H旦glund, R., Ams端ss, C., and F. Palombini, "Observe Notifications as CoAP Multicast Responses", Work in Progress, Internet-Draft, draft-ietf-core-observe-multicast-notifications-14, 22 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-observe-multicast-notifications-14>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. H旦glund, "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>.

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

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[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>.

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[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>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

12.2. Informative References

[I-D.ietf-core-cacheable-oscore]
Ams端ss, C. and M. Tiloca, "End-to-End Protected and Cacheable Responses for the Constrained Application

Protocol (CoAP) using Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-cacheable-oscore-01, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-cacheable-oscore-01>.

[I-D.ietf-core-oscore-capable-proxies] Tiloca, M. and R. H旦glund, "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>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

Appendix A. Document Updates

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

A.1. Version -00 to -01

* Renamed the Multicast-Response-Feedback-Divider Option as Feedback-Divider.

* Fixes in the examples of message exchanges.

* Clarifications and editorial improvements.

A.2. Version -00

* Imported content about proxies from draft-ietf-core-observe-multicast-notifications-12.

Acknowledgments

The authors sincerely thank Carsten Bormann, Klaus Hartke, Jaime Jim迦nez, Matthias Kovatsch, John Preu Mattsson, Jim Schaad, Ludwig Seitz, and Gran Selander for their comments and feedback.

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

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-164 40 Kista
Sweden
Email: marco.tiloca@ri.se

Rikard Hglund
RISE AB
Isafjordsgatan 22
SE-164 40 Kista
Sweden
Email: rikard.hoglund@ri.se

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

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-164 40 Kista
Sweden
Email: francesca.palombini@ericsson.com