CoRE Working Group
Internet-Draft
Updates: 7252, 7641 (if approved)
Intended status: Standards Track
Expires: 3 October 2021

M. Tiloca
R. Hoeglund
RISE AB
C. Amsuess

F. Palombini
Ericsson AB

1 April 2021

# Observe Notifications as CoAP Multicast Responses
# draft-ietf-core-observe-multicast-notifications-00

Abstract

The Constrained Application Protocol (CoAP) allows clients to "observe" resources at a server, and receive notifications as unicast responses upon changes of the resource state. In some use cases, such as based on publish-subscribe, it would be convenient for the server to send a single notification addressed to all the clients observing a same target resource. This document updates RFC7252 and RFC7641, and defines how a server sends observe notifications as response messages over multicast, synchronizing all the observers of a same resource on a same shared Token value. Besides, this document defines how Group OSCORE can be used to protect multicast notifications end-to-end between the server and the observer clients.

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 over IP multicast [I-D.ietf-core-groupcomm-bis]. 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.

However, in a number of use cases, using multicast messages for responses would also be desirable. That is, it would be useful that a server sends observe notifications for a same target resource to multiple observers as responses over IP multicast.

For instance, in CoAP publish-subscribe [I-D.ietf-core-coap-pubsub], multiple clients can subscribe to a topic, by observing the related resource hosted at the responsible broker. When a new value is published on that topic, it would be convenient for the broker to send a single multicast notification at once, to all the subscriber clients observing that topic.

A different use case concerns clients observing a same registration resource at the CoRE Resource Directory [I-D.ietf-core-resource-directory]. For example, multiple clients can benefit of observation for discovering (to-be-created) OSCORE groups [I-D.ietf-core-oscore-groupcomm], by retrieving from the Resource Directory updated links and descriptions to join them through the respective Group Manager [I-D.tiloca-core-oscore-discovery].

More in general, multicast notifications would be beneficial whenever several CoAP clients observe a same target resource at a CoAP server, and can be all notified at once by means of a single response message. However, CoAP does not currently define response messages over IP multicast. This specification fills this gap and provides the following twofold contribution.

First, it updates [RFC7252] and [RFC7641], by defining a method to deliver Observe notifications as CoAP responses addressed to multiple clients, e.g. over IP multicast. In the proposed method, the group of potential observers entrusts the server to manage the Token space for multicast notifications. By doing so, the server provides all the observers of a target resource with the same Token value to bind to their own observation. That Token value is then used in every multicast notification for the target resource. This is achieved by means of an informative unicast response sent by the server to each observer client.

Second, this specification defines how to use Group OSCORE [I-D.ietf-core-oscore-groupcomm] to protect multicast notifications end-to-end between the server and the observer clients. This is also achieved by means of the informative unicast response mentioned above, which additionally includes parameter values used by the server to protect every multicast notification for the target resource by using Group OSCORE. This provides a secure binding between each of such notifications and the observation of each of the clients.

### 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], CBOR [RFC8949], OSCORE [RFC8613], and Group OSCORE [I-D.ietf-core-oscore-groupcomm].

This specification additionally defines the following terminology.

* Traditional observation. A resource observation associated to a single observer client, as defined in [RFC7641].

* Group observation. A resource observation associated to a group of clients. The server sends notifications for the group-observed resource over IP multicast to all the observer clients.

* Phantom request. The CoAP request message that the server would have received to start or cancel a group observation on one of its resources. A phantom request is generated inside the server and does not hit the wire.

* Informative response. A CoAP response message that the server sends to a given client via unicast, providing the client with information on a group observation.

## 2. Server-Side Requirements

The server can, at any time, start a group observation on one of its resources. Practically, the server may want to do that under the following circumstances.

* In the absence of observations for the target resource, the server receives a registration request from a first client wishing to start a traditional observation on that resource.

* When a certain amount of traditional observations has been established on the target resource, the server decides to make those clients part of a group observation on that resource.

The server maintains an observer counter for each group observation to a target resource, as a rough estimation of the observers actively taking part in the group observation.

The server initializes the counter to 0 when starting the group observation, and increments it after a new client starts taking part in that group observation. Also, the server should keep the counter up-to-date over time, for instance by using the method described in Section 6.

This document does not describe a way for the client to influence the server's decision to start group observations. That is done on purpose: the specified mechanism is expected to be used in situations where sending individual notifications is not feasible, or not preferred beyond a certain number of clients observing a target resource. If applications arise where negotiation does make sense, they are welcome to specify additional means to opt in to multicast notifications.

### 2.1. Request

Assuming it is reachable at the address SRV_ADDR and port number SRV_PORT, the server starts a group observation on one of its resources as defined below. The server intends to send multicast notifications for the target resource to the multicast IP address GRP_ADDR and port number GRP_PORT.

1. The server builds a phantom observation request, i.e. a GET request with an Observe option set to 0 (register).

2. The server selects an available value T, from the Token space of a CoAP endpoint used for messages having:

   * As source address and port number, the IP multicast address GRP_ADDR and port number GRP_PORT.

   * As destination address and port number, the server address SRV_ADDR and port number SRV_PORT, intended for accessing the target resource.

   This Token space is under exclusive control of the server.

3. The server processes the phantom observation request above, without transmitting it on the wire. The request is addressed to the resource for which the server wants to start the group observation, as if sent by the group of observers, i.e. with GRP_ADDR as source address and GRP_PORT as source port.

4. Upon processing the self-generated phantom registration request, the server interprets it as an observe registration received from the group of potential observer clients. In particular, from then on, the server MUST use T as its own local Token value associated to that observation, with respect to the (previous hop towards the) clients.

5. The server does not immediately respond to the phantom observation request with a multicast notification sent on the wire. The server stores the phantom observation request as is, throughout the lifetime of the group observation.

6. The server builds a CoAP response message INIT_NOTIF as initial multicast notification for the target resource, in response to the phantom observation request. This message is formatted as other multicast notifications (see Section 2.3) and MUST include the current representation of the target resource as payload. The server stores the message INIT_NOTIF and does not transmit it. The server considers this message as the latest multicast notification for the target resource, until it transmits a new multicast notification for that resource as a CoAP message on the wire. After that, the server deletes the message INIT_NOTIF.

## 2.2. Informative Response

After having started a group observation on a target resource, the server proceeds as follows.

For each traditional observation ongoing on the target resource, the server MAY cancel that observation. Then, the server considers the corresponding clients as now taking part in the group observation, for which it increases the corresponding observer counter accordingly.

The server sends to each of such clients an informative response message, encoded as a unicast response with response code 5.03 (Service Unavailable). As per [RFC7641], such a response does not include an Observe option. The response MUST be Confirmable and MUST NOT encode link-local addresses.

The Content-Format of the informative response is set to application/informative-response+cbor, defined in Section 14.2. The payload of the informative response is a CBOR map including the following parameters, whose CBOR labels are defined in Section 11.

* 'tp_info', with value a CBOR array. This includes the transport-specific information required to correctly receive multicast notifications bound to the phantom observation request. The CBOR array is formatted as defined in Section 2.2.1. This parameter MUST be included.

* 'ph_req', with value the byte serialization of the transport-independent information of the phantom observation request (see Section 2.1), encoded as a CBOR byte string. The value of the CBOR byte string is formatted as defined in Section 2.2.2. This parameter MUST be included.

* 'last_notif', with value the byte serialization of the transport-independent information of the latest multicast notification for the target resource, encoded as a CBOR byte string. The value of the CBOR byte string is formatted as defined in Section 2.2.2. This parameter MAY be included.

The CDDL notation [RFC8610] provided below describes the payload of the informative response.

```
informative_response_payload = {
   1 => array, ; 'tp_info', i.e. transport-specific information
   2 => bstr,  ; 'ph_req' (transport-independent information)
 ? 3 => bstr   ; 'last_notif' (transport-independent information)
}
```

Figure 1: Format of the informative response payload

Upon receiving a registration request to observe the target resource, the server does not create a corresponding individual observation for the requesting client. Instead, the server considers that client as now taking part in the group observation of the target resource, of which it increments the observer counter by 1. Then, the server replies to the client with the same informative response message defined above, which MUST be Confirmable.

Note that this also applies when, with no ongoing traditional observations on the target resource, the server receives a registration request from a first client and decides to start a group observation on the target resource.

### 2.2.1. Encoding of Transport-Specific Message Information

The CBOR array specified in the 'tp_info' parameter is formatted according to the following CDDL notation.

```
tp_info = [
    srv_addr  ; Addressing information of the server
  ? req_info  ; Request data extension
]

srv_addr = (
    tp_id : int,  ; Identifier of the used transport protocol
  + elements      ; Number, format and encoding
                  ; based on the value of 'tp_id'
)

req_info = (
  + elements  ; Number, format and encoding based on
              ; the value of 'tp_id' in 'srv_addr'
)
```

Figure 2: General format of 'tp_info'

The 'srv_addr' element of 'tp_info' specifies the addressing information of the server, and includes at least one element 'tp_id' which is formatted as follows.

* 'tp_id' : this element is a CBOR integer, which specifies the transport protocol used to transport the CoAP response from the server, i.e. a multicast notification in this specification.

  This element takes value from the "Value" column of the "CoAP Transport Information" registry defined in Section 14.4 of this specification. This element MUST be present. The value of this element determines:

  - How many elements are required to follow in 'srv_addr', as well as what information they convey, their encoding and their semantics.

  - How many elements are required in the 'req_info' element of the 'tp_info' array, as well as what information they convey, their encoding and their semantics.

  This specification registers the integer value 1 ("UDP") to be used as value for the 'tp_id' element, when CoAP responses are transported over UDP. In such a case, the full encoding of the 'tp_info' CBOR array is as defined in Section 2.2.1.1.

  Future specifications that consider CoAP multicast notifications transported over different transport protocols MUST:

- Register an entry with an integer value to be used for 'tp_id', in the "CoAP Transport Information" registry defined in Section 14.4 of this specification.

- Accordingly, define the elements of the 'tp_info' CBOR array, i.e. the elements following 'tp_id' in 'srv_addr' as well as the elements in 'req_info', as to what information they convey, their encoding and their semantics.

The 'req_info' element of 'tp_info' specifies transport-specific information related to a pertinent request message, i.e. the phantom observation request in this specification. The exact format of 'req_info' depends on the value of 'tp_id'.

Given a specific value of 'tp_id', the complete set of elements composing 'srv_addr' and 'req_info' in the 'tp_info' CBOR array is indicated by the two columns "Srv Addr" and "Req Info" of the "CoAP Transport Information" registry defined in Section 14.4, respectively.

#### 2.2.1.1. UDP Transport-Specific Information

When CoAP multicast notifications are transported over UDP as per [RFC7252] and [I-D.ietf-core-groupcomm-bis], the server specifies the integer value 1 ("UDP") as value of 'tp_id' in the 'srv_addr' element of the 'tp_info' CBOR array in the error informative response. Then, the rest of the 'tp_info' CBOR array is defined as follows.

* 'srv_addr' includes two more elements following 'tp_id':

  - 'srv_host': this element is a CBOR byte string, with value the destination IP address of the phantom observation request. This parameter is tagged and identified by the CBOR tag 260 "Network Address (IPv4 or IPv6 or MAC Address)". That is, the value of the CBOR byte string is the IP address SRV_ADDR of the server hosting the target resource, from where the server will send multicast notifications for the target resource. This element MUST be present.

  - 'srv_port': this element is a CBOR unsigned integer, with value the destination port number of the phantom observation request. That is, the specified value is the port number SRV_PORT, from where the server will send multicast notifications for the target resource. This element MUST be present.

* 'req_info' includes the following elements:

- 'token': this element is a CBOR byte string, with value the Token value of the phantom observation request generated by the server (see Section 2.1). Note that the same Token value is used for the multicast notifications bound to that phantom observation request (see Section 2.3). This element MUST be present.

- 'cli_addr': this element is a CBOR byte string, with value the source IP address of the phantom observation request. This parameter is tagged and identified by the CBOR tag 260 "Network Address (IPv4 or IPv6 or MAC Address)". That is, the value of the CBOR byte string is the IP multicast address GRP_ADDR, where the server will send multicast notifications for the target resource. This element MUST be present.

- 'cli_port': this element is a CBOR unsigned integer, with value the source port number of the phantom observation request. That is, the specified value is the port number GRP_PORT, where the server will send multicast notifications for the target resource. This element is OPTIONAL. If not included, the default port number 5683 is assumed.

The CDDL notation provided below describes the full 'tp_info' CBOR array using the format above.

```
tp_info = [
    tp_id    : 1,             ; UDP as transport protocol
    srv_host : #6.260(bstr),  ; Src. address of multicast notifications
    srv_port : uint,          ; Src. port of multicast notifications
    token    : bstr,          ; Token of the phantom request and
                              ; associated multicast notifications
    cli_addr : #6.260(bstr),  ; Dst. address of multicast notifications
  ? cli_port : uint           ; Dst. port of multicast notifications
]
```

Figure 3: Format of 'tp_info' with UDP as transport protocol

### 2.2.2. Encoding of Transport-Independent Message Information

For both the parameters 'ph_req' and 'last_notif' in the informative response, the value of the byte string is the concatenation of the following components, in the order specified below.

When defining the value of each component, "CoAP message" refers to the phantom observation request for the 'ph_req' parameter, and to the corresponding latest multicast notification for the 'last_notif' parameter.

* A single byte, with value the content of the Code field in the CoAP message.

* The byte serialization of the complete sequence of CoAP options in the CoAP message.

* If the CoAP message includes a non-zero length payload, the one-byte Payload Marker (0xff) followed by the payload.

## 2.3. Notifications

Upon a change in the status of the target resource under group observation, the server sends a multicast notification, intended to all the clients taking part in the group observation of that resource. In particular, each of such multicast notifications is formatted as follows.

* It MUST be Non-confirmable.

* It MUST include an Observe option, as per [RFC7641].

* It MUST have the same Token value T of the phantom registration request that started the group observation. This Token value is specified in the 'token' element of 'req_info' under the 'tp_info' parameter, in the informative response message sent to all the observer clients.

  That is, every multicast notification for a target resource is not bound to the observation requests from the different clients, but rather to the phantom registration request associated to the whole set of clients taking part in the group observation of that resource.

* It MUST be sent from the same IP address SRV_ADDR and port number SRV_PORT where: i) the original Observe registration requests are sent to by the clients; and ii) the corresponding informative responses are sent from by the server (see Section 2.2). These are indicated to the observer clients as value of the 'srv_host' and 'srv_port' elements of 'srv_addr' under the 'tp_info' parameter, in the informative response message (see Section 2.2.1.1). That is, redirection MUST NOT be used.

* It MUST be sent to the IP multicast address GRP_ADDR and port number GRP_PORT. These are indicated to the observer clients as value of the 'cli_addr' and 'cli_port' elements of 'req_info' under the 'tp_info' parameter, in the informative response message (see Section 2.2.1.1).

For each target resource with an active group observation, the server MUST store the latest multicast notification.

## 2.4. Congestion Control

In order to not cause congestion, the server should conservatively control the sending of multicast notifications. In particular:

* The multicast notifications MUST be Non-confirmable.

* In constrained environments such as low-power, lossy networks (LLNs), the server should only support multicast notifications for resources that are small. Following related guidelines from Section 2.2.4 of [I-D.ietf-core-groupcomm-bis], this can consist, for example, in having the payload of multicast notifications as limited to approximately 5% of the IP Maximum Transmit Unit (MTU) size, so that it fits into a single link-layer frame in case IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) (see Section 4 of [RFC4944]) is used.

* The server SHOULD provide multicast notifications with the smallest possible IP multicast scope that fulfills the application needs. For example, following related guidelines from Section 2.2.4 of [I-D.ietf-core-groupcomm-bis], site-local scope is always preferred over global scope IP multicast, if this fulfills the application needs. Similarly, realm-local scope is always preferred over site-local scope, if this fulfills the application needs.

* Following related guidelines from Section 4.5.1 of [RFC7641], the server SHOULD NOT send more than one multicast notification every 3 seconds, and SHOULD use an even less aggressive rate when possible (see also Section 3.1.2 of [RFC8085]). The transmission rate of multicast notifications should also take into account the avoidance of a possible "broadcast storm" problem [MOBICOM99]. This prevents a following, considerable increase of the channel load, whose origin would be likely attributed to a router rather than the server.

## 2.5. Cancellation

At any point in time, the server may want to cancel a group observation of a target resource. For instance, the server may realize that no clients or not enough clients are interested in taking part in the group observation anymore. A possible approach that the server can use to assess this is defined in Section 6.

In order to cancel the group observation, the server sends to itself a phantom cancellation request, i.e. a GET request with an Observe option set to 1 (deregister), without transmitting it on the wire. As per Section 3.6 of [RFC7641], all other options MUST be identical to those in the phantom registration request, except for the set of ETag Options. This request has the same Token value T of the phantom registration request, and is addressed to the resource for which the server wants to end the group observation, as if sent by the group of observers, i.e. with the multicast IP address GRP_ADDR as source address and the port number GRP_PORT as source port.

After that, the server sends a multicast response with response code 5.03 (Service Unavailable), signaling that the group observation has been terminated. The response has no payload, and is sent to the same multicast IP address GRP_ADDR and port number GRP_PORT used to send the multicast notifications related to the target resource. As per [RFC7641], this response does not include an Observe option. Finally, the server releases the resources allocated for the group observation, and especially frees up the Token value T used at its CoAP endpoint.

# 3. Client-Side Requirements

## 3.1. Request

A client sends an observation request to the server as described in [RFC7641], i.e. a GET request with an Observe option set to 0 (register). The request MUST NOT encode link-local addresses. If the server is not configured to accept registrations on that target resource with a group observation, this would still result in a positive notification response to the client as described in [RFC7641].

## 3.2. Informative Response

Upon receiving the informative response defined in Section 2.2, the client proceeds as follows.

1. The client configures an observation of the target resource. To this end, it relies on a CoAP endpoint used for messages having:

   * As source address and port number, the server address SRV_ADDR and port number SRV_PORT intended for accessing the target resource. These are specified as value of the 'srv_host' and 'srv_port' elements of 'srv_addr' under the 'tp_info' parameter, in the informative response (see Section 2.2.1.1).

* As destination address and port number, the IP multicast address GRP_ADDR and port number GRP_PORT. These are specified as value of the 'cli_addr' and 'cli_port' elements of 'req_info' under the 'tp_info' parameter, in the informative response (see Section 2.2.1.1). If the 'cli_port' element is omitted in 'req_info', the client MUST assume the default port number 5683 as GRP_PORT.

2. The client rebuilds the phantom registration request, by using:

   * The transport-independent information, specified in the 'ph_req' parameter of the informative response.

   * The Token value T, specified in the 'token' element of 'req_info' under the 'tp_info' parameter of the informative response.

3. The client stores the phantom registration request, as associated to the observation of the target resource. In particular, the client MUST use the Token value T of this phantom registration request as its own local Token value associated to that group observation, with respect to the server. The particular way to achieve this is implementation specific.

4. If the informative response includes the parameter 'last_notif', the client rebuilds the latest multicast notification, by using:

   * The transport-independent information, specified in the 'last_notif' parameter of the informative response.

   * The Token value T, specified in the 'token' element of 'req_info' under the 'tp_info' parameter of the informative response.

5. If the informative response includes the parameter 'last_notif', the client processes the multicast notification rebuilt in step 4 as defined in Section 3.2 of [RFC7641]. In particular, the value of the Observe option is used as initial baseline for notification reordering in this group observation.

6. If a traditional observation to the target resource is ongoing, the client MAY silently cancel it without notifying the server.

If any of the expected fields in the informative response are not present or malformed, the client MAY try sending a new registration request to the server (see Section 3.1). Otherwise, the client SHOULD explicitly withdraw from the group observation.

Appendix A describes possible alternative ways for clients to retrieve the phantom registration request and other information related to a group observation.

### 3.3. Notifications

After having successfully processed the informative response as defined in Section 3.2, the client will receive, accept and process multicast notifications about the state of the target resource from the server, as responses to the phantom registration request and with Token value T.

The client relies on the value of the Observe option for notification reordering, as defined in Section 3.4 of [RFC7641].

### 3.4. Cancellation

At a certain point in time, a client may become not interested in receiving further multicast notifications about a target resource. When this happens, the client can simply "forget" about being part of the group observation for that target resource, as per Section 3.6 of [RFC7641].

When, later on, the server sends the next multicast notification, the client will not recognize the Token value T in the message. Since the multicast notification is Non-confirmable, it is OPTIONAL for the client to reject the multicast notification with a Reset message, as defined in Section 3.5 of [RFC7641].

In case the server has canceled a group observation as defined in Section 2.5, the client simply forgets about the group observation and frees up the used Token value T for that endpoint, upon receiving the multicast error response defined in Section 2.5.

## 4. Web Linking

The possible use of multicast notifications in a group observation may be indicated by a target "grp_obs" attribute in a web link [RFC8288] to a resource, e.g. using a link-format document [RFC6690] if the resource is accessible over CoAP.

The "grp_obs" attribute is a hint, indicating that the server might send multicast notifications for observations of the resource targeted by the link. Note that this is simply a hint, i.e. it does not include any information required to participate in a group observation, and to receive and process multicast notifications.

A value MUST NOT be given for the "grp_obs" attribute; any present value MUST be ignored by parsers. The "grp_obs" attribute MUST NOT appear more than once in a given link-value; occurrences after the first MUST be ignored by parsers.

The example in Figure 4 shows a use of the "grp_obs" attribute: the client does resource discovery on a server and gets back a list of resources, one of which includes the "grp_obs" attribute indicating that the server might send multicast notifications for observations of that resource. The link-format notation (see Section 5 of [RFC6690]) is used.

```
REQ: GET /.well-known/core

RES: 2.05 Content
    </sensors/temp>;grp_obs,
    </sensors/light>;if="sensor"
```

Figure 4: The Web Link

## 5. Example

The following example refers to two clients C_1 and C_2 that 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".

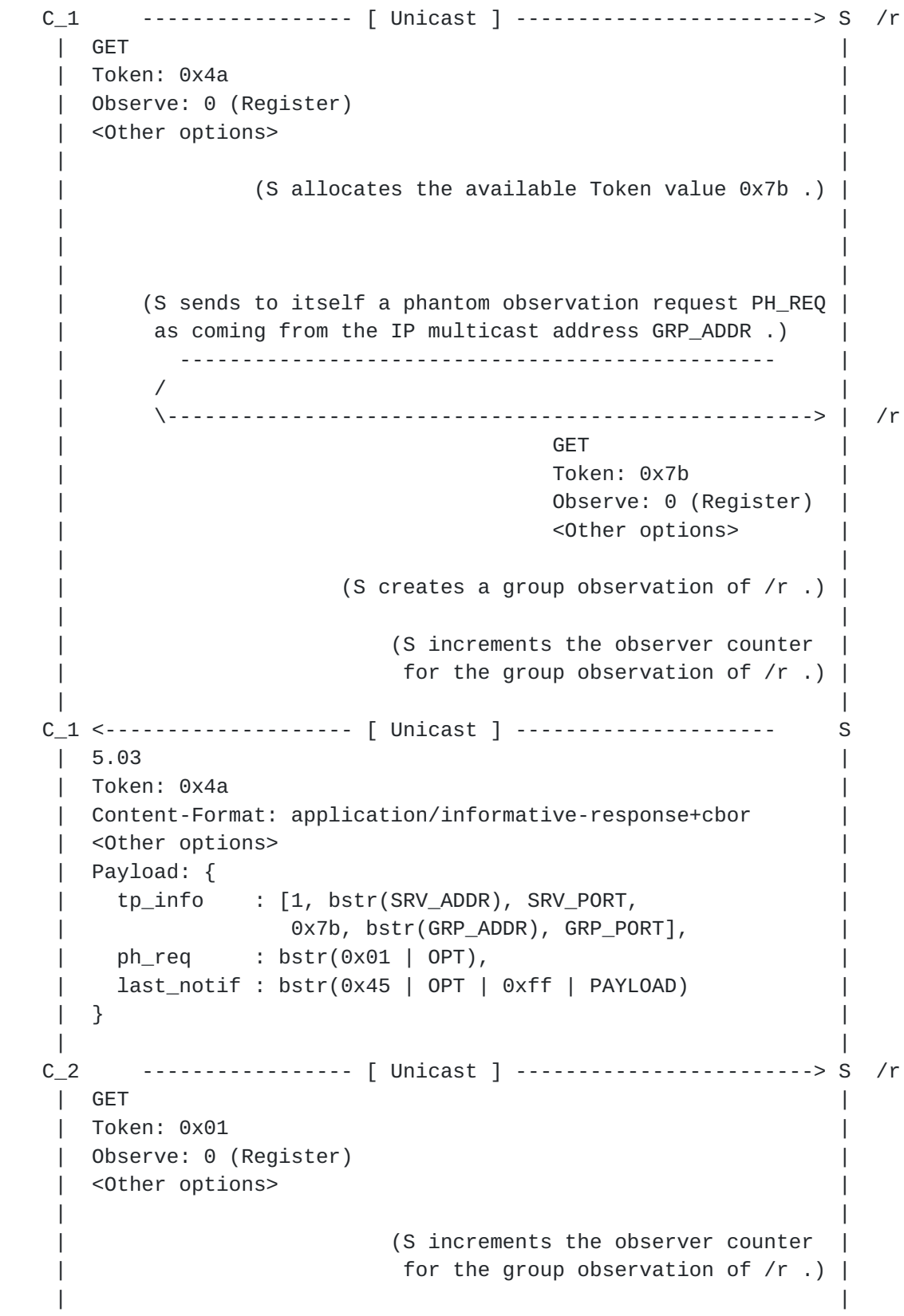
The server S sends multicast notifications to the IP multicast address GRP_ADDR and port number GRP_PORT, and starts the group observation upon receiving a registration request from a first client that wishes to start a traditional observation on the resource /r.

The following notation is used for the payload of the informative responses:

* 'bstr(X)' denotes a CBOR byte string with value the byte serialization of X, with '|' denoting byte concatenation.

* 'OPT' denotes a sequence of CoAP options. This refers to the phantom registration request encoded by the 'ph_req' parameter, or to the corresponding latest multicast notification encoded by the 'last_notif' parameter.

* 'PAYLOAD' denotes a CoAP payload. This refers to the latest multicast notification encoded by the 'last_notif' parameter.


C_1
[ Unicast ]
S
/r
GET
Token: 0x4a
Observe: 0 (Register)
<Other 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 .)
/r
GET
Token: 0x7b
Observe: 0 (Register)
<Other options>
(S creates a group observation of /r .)
(S increments the observer counter
for the group observation of /r .)
C_1
[ Unicast ]
S
5.03
Token: 0x4a
Content-Format: application/informative-response+cbor
<Other options>
Payload: {
tp_info : [1, bstr(SRV_ADDR), SRV_PORT,
0x7b, bstr(GRP_ADDR), GRP_PORT],
ph_req : bstr(0x01 | OPT),
last_notif : bstr(0x45 | OPT | 0xff | PAYLOAD)
}
C_2
[ Unicast ]
S
/r
GET
Token: 0x01
Observe: 0 (Register)
<Other options>
(S increments the observer counter
for the group observation of /r .)

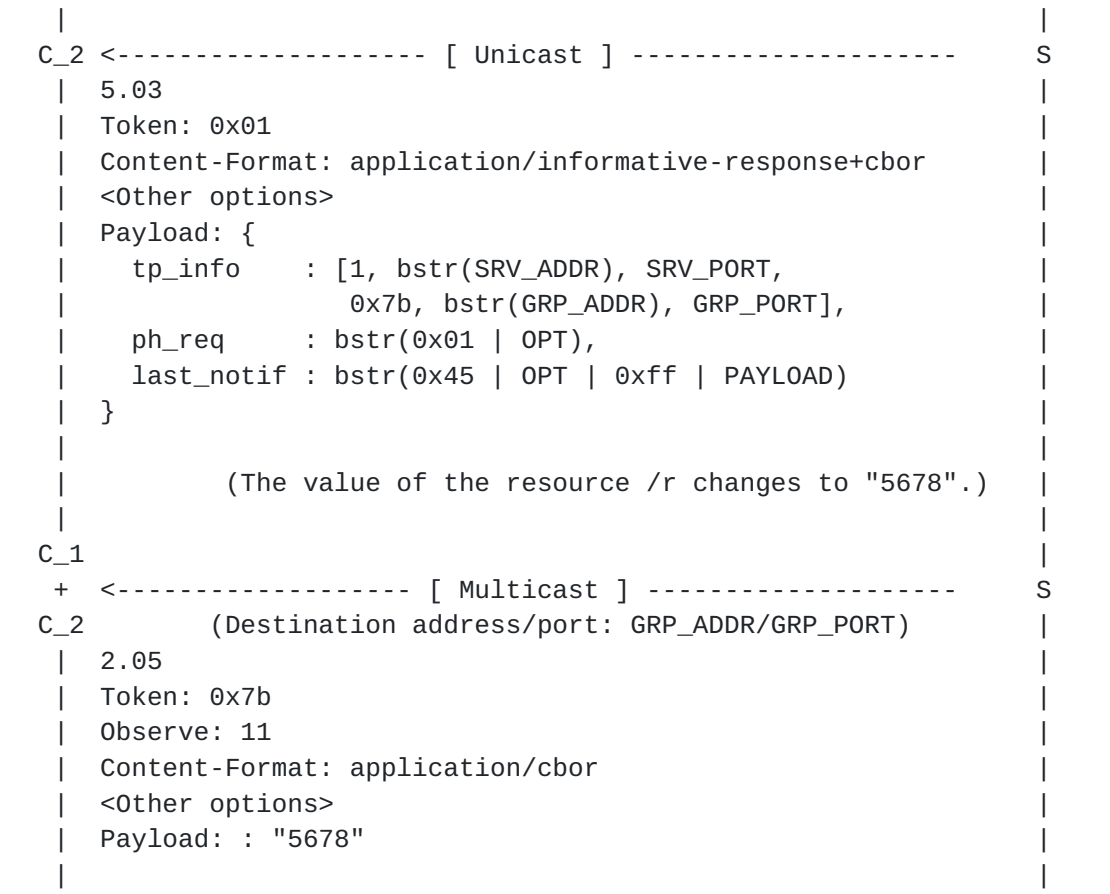



Figure 5: Example of group observation

## 6. Rough Counting of Clients in the Group Observation

To allow the server to keep an estimate of interested clients without creating undue traffic on the network, a new CoAP option is introduced, which SHOULD be supported by clients that listen to multicast responses.

The option is called Multicast-Response-Feedback-Divider. As summarized in Figure 6, the option is not critical but proxy-unsafe, and integer valued.

| No. | C | U | N | R | Name | Format | Len. | Default |
|---|---|---|---|---|---|---|---|---|
| TBD | | x | | | Multicast-Response-Feedback-Divider | uint | 0-1 | (none) |

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

Figure 6: Multicast-Response-Feedback-Divider

The Multicast-Response-Feedback-Divider option is of class E for OSCORE [RFC8613][I-D.ietf-core-oscore-groupcomm].

### 6.1. Processing on the Client Side

Upon receiving a response with a Multicast-Response-Feedback-Divider option, a client SHOULD acknowledge its interest in continuing receiving multicast notifications for the target resource, as described below.

The client picks an integer random number I, from 0 inclusive to the number Z = (2 ** Q) exclusive, where Q is the value specified in the option and "**" is the exponentiation operator. If I is different than 0, the client takes no further action. Otherwise, the client should wait a random fraction of the Leisure time (see Section 8.2 of [RFC7252]), and then registers a regular unicast observation on the same target resource.

To this end, the client essentially follows the steps that got it originally subscribed to group notifications for the target resource. In particular, the client sends an observation request to the server, i.e. a GET request with an Observe option set to 0 (register). The request MUST be addressed to the same target resource, and MUST have the same destination IP address and port number used for the original registration request, regardless the source IP address and port number of the received multicast notification.

Since the observation registration is only done for its side effect of showing as an attempted observation at the server, the client MUST send the unicast request in a non confirmable way, and with the maximum No-Response setting [RFC7967]. In the request, the client MUST include a Multicast-Response-Feedback-Divider option, whose value MUST be empty (Option Length = 0). The client does not need to wait for responses, and can keep processing further notifications on the same token.

The client MUST ignore the Multicast-Response-Feedback-Divider option, if the multicast notification is retrieved from the 'last_notif' parameter of an informative response (see Section 2.2). A client includes the Multicast-Response-Feedback-Divider option only in a re-registration request triggered by the server as described above, and MUST NOT include it in any other request.

As the Multicast-Response-Feedback-Divider option is unsafe to forward, a proxy needs to answer it on its own, and is later counted as a single client.

Appendix B.1 provides a description in pseudo-code of the operations above performed by the client.

### 6.2. Processing on the Server Side

In order to avoid needless use of network resources, a server SHOULD keep a rough, updated count of the number of clients taking part in the group observation of a target resource. To this end, the server updates the value COUNT of the associated observer counter (see Section 2), for instance by using the method described below.

#### 6.2.1. Request for Feedback

When it wants to obtain a new estimated count, the server considers a number M of confirmations it would like to receive from the clients. It is up to applications to define policies about how the server determines and possibly adjusts the value of M.

Then, the server computes the value Q = max(L, 0), where:

* L is computed as L = ceil(log2(N / M)).

* N is the current value of the observer counter, possibly rounded up to 1, i.e. N = max(COUNT, 1).

Finally, the server sets Q as the value of the Multicast-Response-Feedback-Divider option, which is sent within a successful multicast notification.

If several multicast notifications are sent in a burst fashion, it is RECOMMENDED for the server to include the Multicast-Response-Feedback-Divider option only in the first one of those notifications.

#### 6.2.2. Collection of Feedback

The server collects unicast observation requests from the clients, for an amount of time of MAX_CONFIRMATION_WAIT seconds. During this time, the server regularly increments the observer counter when adding a new client to the group observation (see Section 2.2).

It is up to applications to define the value of MAX_CONFIRMATION_WAIT, which has to take into account the transmission time of the multicast notification and of unicast observation requests, as well as the leisure time of the clients, which may be hard to know or estimate for the server.

If this information is not known to the server, it is recommended to define MAX_CONFIRMATION_WAIT as follows.

MAX_CONFIRMATION_WAIT = MAX_RTT + MAX_CLIENT_REQUEST_DELAY

where MAX_RTT is as defined in Section 4.8.2 of [RFC7252] and has default value 202 seconds, while MAX_CLIENT_REQUEST_DELAY is equivalent to MAX_SERVER_RESPONSE_DELAY defined in Section 2.3.1 of [I-D.ietf-core-groupcomm-bis] and has default value 250 seconds. In the absence of more specific information, the server can thus consider a conservative MAX_CONFIRMATION_WAIT of 452 seconds.

If more information is available in deployments, a much shorter MAX_CONFIRMATION_WAIT can be set. This can be based on a realistic round trip time (replacing MAX_RTT) and on the largest leisure time configured on the clients (replacing MAX_CLIENT_REQUEST_DELAY), e.g. DEFAULT_LEISURE = 5 seconds, thus shortening MAX_CONFIRMATION_WAIT to a few seconds.

#### 6.2.3. Processing of Feedback

Once MAX_CONFIRMATION_WAIT seconds have passed, the server counts the R confirmations arrived as unicast observation requests to the target resource, since the multicast notification with the Multicast-Response-Feedback-Divider option has been sent. In particular, the server considers a unicast observation request as a confirmation from a client only if it includes a Multicast-Response-Feedback-Divider option with an empty value (Option Length = 0).

Then, the server computes a feedback indicator as E = R * (2 ** Q), where "**" is the exponentiation operator. According to what defined by application policies, the server determines the next time when to ask clients for their confirmation, e.g. after a certain number of multicast notifications has been sent. For example, the decision can be influenced by the reception of no confirmations from the clients, i.e. R = 0, or by the value of the ratios (E/N) and (N/E).

Finally, the server computes a new estimated count of the observers. To this end the server first consider COUNT' as the current value of the observer counter at this point in time. Note that COUNT' may be greater than the value COUNT used at the beginning of this process, if the server has incremented the observer counter upon adding new clients to the group observation (see Section 2.2).

In particular, the server computes the new estimated count value as COUNT' + ((E - N) / D), where D > 0 is an integer value used as dampener. This step has to be performed atomically. That is, until this step is completed, the server MUST hold the processing of an observation request for the same target resource from a new client. Finally, the server considers the result as the current observer counter, and assesses it for possibly canceling the group observation (see Section 2.5).

This estimate is skewed by packet loss, but it gives the server a sufficiently good estimation for further counts and for deciding when to cancel the group observation. It is up to applications to define policies about how the server takes the newly updated estimate into account and determines whether to cancel the group observation.

As an example, if the server currently estimates that N = COUNT = 32 observers are active and considers a constant M = 8, it sends out a notification with Multicast-Response-Feedback-Divider: 2. Then, out of 18 actually active clients, 5 send a re-registration request based on their random draw, of which one request gets lost, thus leaving 4 re-registration requests received by the server. Also, no new clients have been added to the group observation during this time, i.e. COUNT' is equal to COUNT. As a consequence, assuming that a dampener value D = 1 is used, the server computes the new estimated count value as 32 + (16 - 32) = 16, and keeps the group observation active.

To produce a most accurate updated counter, a server can include a Multicast-Response-Feedback-Divider option with value Q = 0 in its multicast notifications, as if M is equal to N. This will trigger all the active clients to state their interest in continuing receiving notifications for the target resource. Thus, the amount R of arrived confirmations is affected only by possible packet loss.

Appendix B.3 provides a description in pseudo-code of the operations above performed by the server, including example behaviors for scheduling the next count update and deciding whether to cancel the group observation.

## 7. Protection of Multicast Notifications with Group OSCORE

A server can protect multicast notifications by using Group OSCORE [I-D.ietf-core-oscore-groupcomm], thus ensuring they are protected end-to-end with the observer clients. This requires that both the server and the clients interested in receiving multicast notifications from that server are members of the same OSCORE group.

In some settings, the OSCORE group to refer to can be pre-configured on the clients and the server. In such a case, a server which is aware of such pre-configuration can simply assume a client to be already member of the correct OSCORE group.

In any other case, the server MAY communicate to clients what OSCORE group they are required to join, by providing additional guidance in the informative response as described in Section 7.1. Note that clients can already be members of the right OSCORE group, in case they have previously joined it to securely communicate with the same and/or other servers to access their resources.

Both the clients and the server MAY join the OSCORE group by using the approach described in [I-D.ietf-ace-key-groupcomm-oscore] and based on the ACE framework for Authentication and Authorization in constrained environments [I-D.ietf-ace-oauth-authz]. Further details on how to discover the OSCORE group and join it are out of the scope of this specification.

If multicast notifications are protected using Group OSCORE, the original registration requests and related unicast (notification) responses MUST also be secured, including and especially the informative responses from the server.

To this end, alternative security protocols than Group OSCORE, such as OSCORE [RFC8613] and/or DTLS [RFC6347][I-D.ietf-tls-dtls13], can be used to protect other exchanges via unicast between the server and each client, including the original client registration (see Section 3).

### 7.1. Signaling the OSCORE Group in the Informative Response

This section describes a mechanism for the server to communicate to the client what OSCORE group to join in order to decrypt and verify the multicast notifications protected with group OSCORE. The client MAY use the information provided by the server to start the ACE joining procedure described in [I-D.ietf-ace-key-groupcomm-oscore]. This mechanism is OPTIONAL to support for the client and server.

Additionally to what defined in Section 2, the CBOR map in the informative response payload contains the following fields, whose CBOR labels are defined in Section 11.

* 'join_uri', with value the URI for joining the OSCORE group at the respective Group Manager, encoded as a CBOR text string. If the procedure described in [I-D.ietf-ace-key-groupcomm-oscore] is used for joining, this field specifically indicates the URI of the group-membership resource at the Group Manager.

* 'sec_gp', with value the name of the OSCORE group, encoded as a CBOR text string.

* Optionally, 'as_uri', with value the URI of the Authorization Server associated to the Group Manager for the OSCORE group, encoded as a CBOR text string.

* Optionally, 'cs_alg', with value the COSE algorithm [I-D.ietf-cose-rfc8152bis-algs] used to countersign messages in the OSCORE group, encoded as a CBOR text string or integer. The value is taken from the 'Value' column of the "COSE Algorithms" registry [COSE.Algorithms].

* Optionally, 'cs_alg_crv', with value the elliptic curve (if applicable) for the COSE algorithm [I-D.ietf-cose-rfc8152bis-algs] used to countersign messages in the OSCORE group, encoded as a CBOR text string or integer. The value is taken from the 'Value' column of the "COSE Elliptic Curve" registry [COSE.Elliptic.Curves].

* Optionally, 'cs_key_kty', with value the COSE key type [I-D.ietf-cose-rfc8152bis-struct] of countersignature keys used to countersign messages in the OSCORE group, encoded as a CBOR text string or an integer. The value is taken from the 'Value' column of the "COSE Key Types" registry [COSE.Key.Types].

* Optionally, 'cs_key_crv', with value the elliptic curve (if applicable) of countersignature keys used to countersign messages in the OSCORE group, encoded as a CBOR text string or integer. The value is taken from the 'Value' column of the "COSE Elliptic Curve" registry [COSE.Elliptic.Curves].

* Optionally, 'cs_kenc', with value the encoding of the public keys used in the OSCORE group, encoded as a CBOR integer. The value is taken from the 'Confirmation Key' column of the "CWT Confirmation Method" registry defined in [RFC8747]. Future specifications may define additional values for this parameter.

* Optionally, 'alg', with value the COSE AEAD algorithm [I-D.ietf-cose-rfc8152bis-algs], encoded as a CBOR text string or integer. The value is taken from the 'Value' column of the "COSE Algorithms" registry [COSE.Algorithms].

* Optionally, 'hkdf', with value the COSE HKDF algorithm [I-D.ietf-cose-rfc8152bis-algs], encoded as a CBOR text string or integer. The value is taken from the 'Value' column of the "COSE Algorithms" registry [COSE.Algorithms].

The values of 'cs_alg', 'cs_alg_crv', 'cs_key_kty', 'cs_key_crv' and 'cs_key_kenc' provide an early knowledge of the format and encoding of public keys used in the OSCORE group. Thus, the client does not need to ask the Group Manager for this information as a preliminary step before the (ACE) join process, or to perform a trial-and-error exchange with the Group Manager upon joining the group. Hence, the client is able to provide the Group Manager with its own public key in the correct expected format and encoding, at the very first step of the (ACE) join process.

The values of 'cs_alg', 'alg' and 'hkdf' provide an early knowledge of the algorithms used in the OSCORE group. Thus, the client is able to decide whether to actually proceed with the (ACE) join process, depending on its support for the indicated algorithms.

As mentioned above, since this mechanism is OPTIONAL, all the fields are OPTIONAL in the informative response. However, the 'join_uri' and 'sec_gp' fields MUST be present if the mechanism is implemented and used. If any of the fields are present without the 'join_uri' and 'sec_gp' fields present, the client MUST ignore these fields, since they would not be sufficient to start the (ACE) join procedure. When this happens, the client MAY try sending a new registration request to the server (see Section 3.1). Otherwise, the client SHOULD explicitly withdraw from the group observation.

Appendix C describes a possible alternative approach, where the server self-manages the OSCORE group, and provides the observer clients with the necessary keying material in the informative response. The approach in Appendix C MUST NOT be used together with the mechanism defined in this section for indicating what OSCORE group to join.

## 7.2. Server-Side Requirements

When using Group OSCORE to protect multicast notifications, the server performs the operations described in Section 2, with the following differences.

### 7.2.1. Registration

The phantom registration request MUST be secured, by using Group OSCORE. In particular, the group mode of Group OSCORE defined in Section 8 of [I-D.ietf-core-oscore-groupcomm] MUST be used.

The server protects the phantom registration request as defined in Section 8.1 of [I-D.ietf-core-oscore-groupcomm], as if it was the actual sender, i.e. by using its Sender Context. As a consequence, the server consumes the current value of its Sender Sequence Number

SN in the OSCORE group, and hence updates it to SN* = (SN + 1). Consistently, the OSCORE option in the phantom registration request includes:

* As 'kid', the Sender ID of the server in the OSCORE group.

* As 'piv', the previously consumed Sender Sequence Number value SN of the server in the OSCORE group, i.e. (SN* - 1).

### 7.2.2. Informative Response

The value of the CBOR byte string in the 'ph_req' parameter encodes the phantom observation request as a message protected with Group OSCORE (see Section 7.2.1). As a consequence: the specified Code is always 0.05 (FETCH); the sequence of CoAP options will be limited to the outer, non encrypted options; a payload is always present, as the authenticated ciphertext followed by the counter signature.

Similarly, the value of the CBOR byte string in the 'last_notif' parameter encodes the latest multicast notification as a message protected with Group OSCORE (see Section 7.2.3). This applies also to the initial multicast notification INIT_NOTIF built in step 6 of Section 2.1.

Optionally, the informative response includes information on the OSCORE group to join, as additional parameters (see Section 7.1).

### 7.2.3. Notifications

The server MUST protect every multicast notification for the target resource with Group OSCORE. In particular, the group mode of Group OSCORE defined in Section 8 of [I-D.ietf-core-oscore-groupcomm] MUST be used.

The process described in Section 8.3 of [I-D.ietf-core-oscore-groupcomm] applies, with the following additions when building the two OSCORE 'external_aad' to encrypt and countersign the multicast notification (see Sections 4.3.1 and 4.3.2 of [I-D.ietf-core-oscore-groupcomm]).

* The 'request_kid' is the 'kid' value in the OSCORE option of the phantom registration request, i.e. the Sender ID of the server.

* The 'request_piv' is the 'piv' value in the OSCORE option of the phantom registration request, i.e. the consumed Sender Sequence Number SN of the server.

* The 'request_kid_context' is the 'kid context' value in the OSCORE option of the phantom registration request, i.e. the Group Identifier value (Gid) of the OSCORE group used as ID Context.

Note that these same values are used to protect each and every multicast notification sent for the target resource under this group observation.

#### 7.2.4. Cancellation

When canceling a group observation (see Section 2.5), the phantom cancellation request MUST be secured, by using Group OSCORE. In particular, the group mode of Group OSCORE defined in Section 8 of [I-D.ietf-core-oscore-groupcomm] MUST be used.

Like defined in Section 7.2.1 for the phantom registration request, the server protects the phantom cancellation request as per Section 8.1 of [I-D.ietf-core-oscore-groupcomm], by using its Sender Context and consuming its current Sender Sequence number in the OSCORE group, from its Sender Context. The following, corresponding multicast error response defined in Section 2.5 is also protected with Group OSCORE, as per Section 8.3 of [I-D.ietf-core-oscore-groupcomm].

Note that, differently from the multicast notifications, this multicast error response will be the only one securely paired with the phantom cancellation request.

### 7.3. Client-Side Requirements

When using Group OSCORE to protect multicast notifications, the client performs as described in Section 3, with the following differences.

#### 7.3.1. Informative Response

Upon receiving the informative response from the server, the client performs as described in Section 3.2, with the following additions.

Once completed step 2, the client decrypts and verifies the rebuilt phantom registration request as defined in Section 8.2 of [I-D.ietf-core-oscore-groupcomm], with the following differences.

* The client MUST NOT perform any replay check. That is, the client skips step 3 in Section 8.2 of [RFC8613].

* If decryption and verification of the phantom registration request succeed:

- The client MUST NOT update the Replay Window in the Recipient Context associated to the server. That is, the client skips the second bullet of step 6 in Section 8.2 of [RFC8613].

- The client MUST NOT take any further process as normally expected according to [RFC7252]. That is, the client skips step 8 in Section 8.2 of [RFC8613]. In particular, the client MUST NOT deliver the phantom registration request to the application, and MUST NOT take any action in the Token space of its unicast endpoint, where the informative response has been received.

- The client stores the values of the 'kid', 'piv' and 'kid context' fields from the OSCORE option of the phantom registration request.

* If decryption and verification of the phantom registration request fail, the client MAY try sending a new registration request to the server (see Section 3.1). Otherwise, the client SHOULD explicitly withdraw from the group observation.

* If the informative response includes the parameter 'last_notif', the client also decrypts and verifies the latest multicast notification rebuilt in step 4, just like it would for the multicast notifications transmitted as CoAP messages on the wire (see Section 7.3.2). The client proceeds with step 5 if decryption and verification of the latest multicast notification succeed, or to step 6 otherwise.

### 7.3.2. Notifications

After having successfully processed the informative response as defined in Section 7.3.1, the client will decrypt and verify every multicast notification for the target resource as defined in Section 8.4 of [I-D.ietf-core-oscore-groupcomm], with the following difference.

The client MUST set the two 'external_aad' defined in Sections 4.3.1 and 4.3.2 of [I-D.ietf-core-oscore-groupcomm] as follows. The particular way to achieve this is implementation specific.

* 'request_kid' takes the value of the 'kid' field from the OSCORE option of the phantom registration request (see Section 7.3.1).

* 'request_piv' takes the value of the 'piv' field from the OSCORE option of the phantom registration request (see Section 7.3.1).

* 'request_kid_context' takes the value of the 'kid context' field from the OSCORE option of the phantom registration request (see Section 7.3.1).

Note that these same values are used to decrypt and verify each and every multicast notification received for the target resource.

The replay protection and checking of multicast notifications is performed as specified in Section 4.1.3.5.2 of [RFC8613].

## 8. Example with Group OSCORE

The following example refers to two clients C_1 and C_2 that 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 upon receiving a registration request from a first client that wishes to start a traditional observation on the resource /r.

Pairwise communication over unicast is protected with OSCORE, while S protects multicast notifications with Group OSCORE. Specifically:

* C_1 and S have a pairwise OSCORE Security Context. In particular, C_1 has 'kid' = 1 as Sender ID, and SN_1 = 101 as Sender Sequence Number. Also, S has 'kid' = 3 as Sender ID, and SN_3 = 301 as Sender Sequence Number.

* C_2 and S have a pairwise OSCORE Security Context. In particular, C_2 has 'kid' = 2 as Sender ID, and SN_2 = 201 as Sender Sequence Number. Also, S has 'kid' = 4 as Sender ID, and SN_4 = 401 as Sender Sequence Number.

* S is a member of the OSCORE group with name "myGroup", and 'kid context' = 0x57ab2e as Group ID. In the OSCORE group, S has 'kid' = 5 as Sender ID, and SN_5 = 501 as Sender Sequence Number.

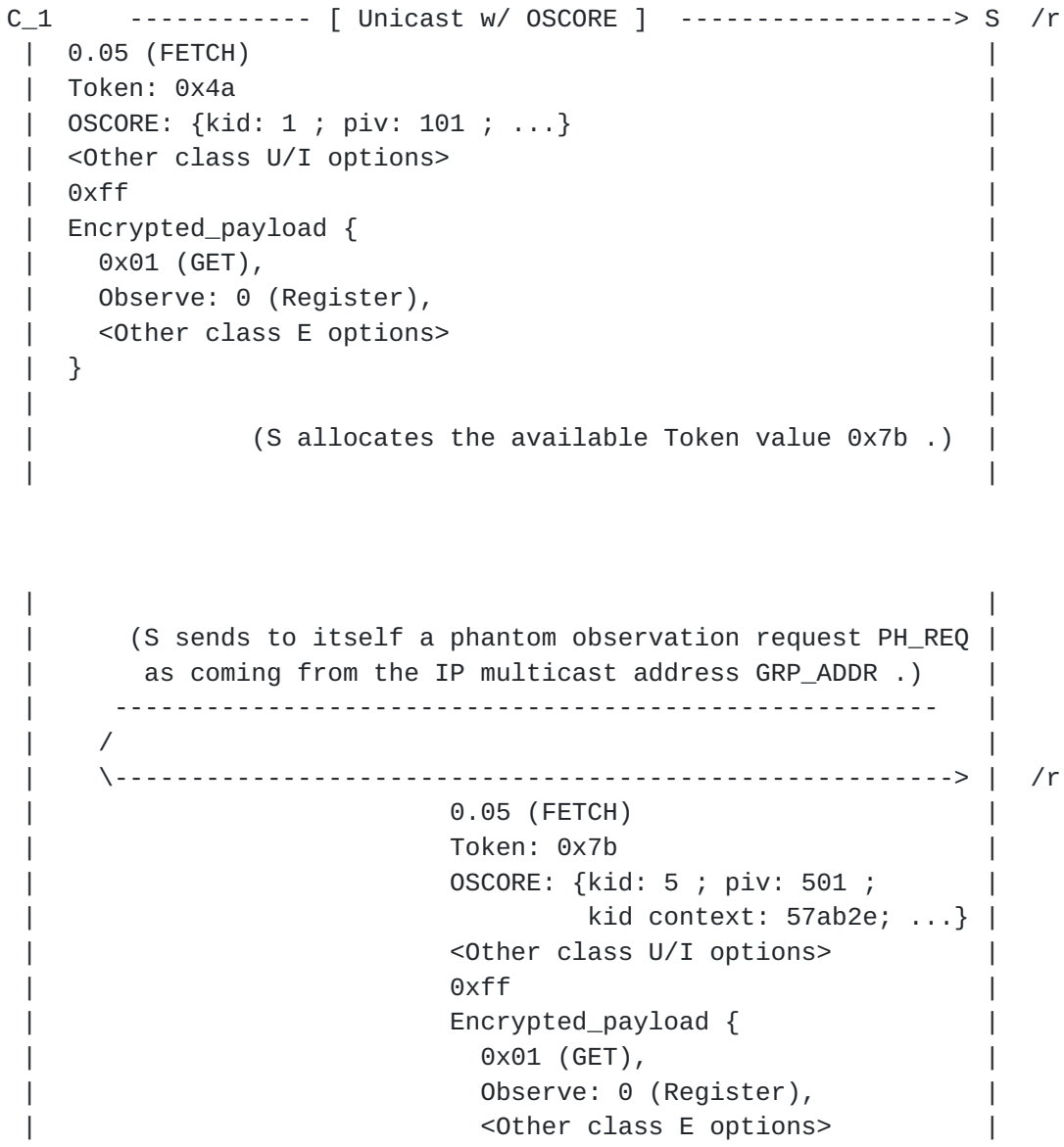
The following notation is used for the payload of the informative responses:

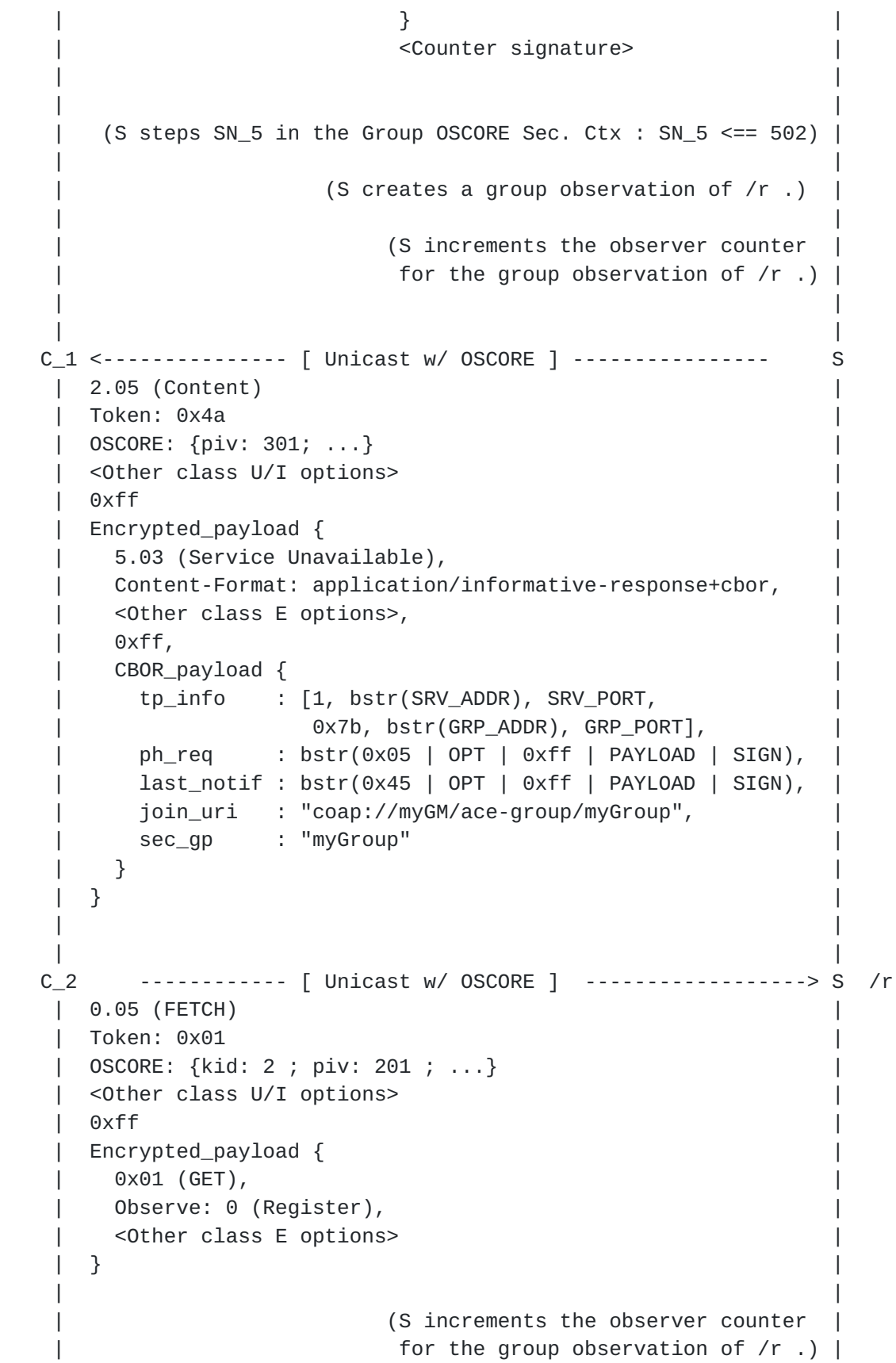
* 'bstr(X)' denotes a CBOR byte string with value the byte serialization of X, with '|' denoting byte concatenation.

* 'OPT' denotes a sequence of CoAP options. This refers to the phantom registration request encoded by the 'ph_req' parameter, or to the corresponding latest multicast notification encoded by the 'last_notif' parameter.

* 'PAYLOAD' denotes an encrypted CoAP payload. This refers to the phantom registration request encoded by the 'ph_req' parameter, or to the corresponding latest multicast notification encoded by the 'last_notif' parameter.

* 'SIGN' denotes the counter signature appended to an encrypted CoAP payload. This refers to the phantom registration request encoded by the 'ph_req' parameter, or to the corresponding latest multicast notification encoded by the 'last_notif' parameter.




}
<Counter signature>
(S steps SN_5 in the Group OSCORE Sec. Ctx : SN_5 <== 502)
(S creates a group observation of /r .)
(S increments the observer counter for the group observation of /r .)
C_1
S
[ Unicast w/ OSCORE ]
2.05 (Content)
Token: 0x4a
OSCORE: {piv: 301; ...}
<Other class U/I options>
0xff
Encrypted_payload {
5.03 (Service Unavailable),
Content-Format: application/informative-response+cbor,
<Other class E options>,
0xff,
CBOR_payload {
tp_info : [1, bstr(SRV_ADDR), SRV_PORT,
0x7b, bstr(GRP_ADDR), GRP_PORT],
ph_req : bstr(0x05 | OPT | 0xff | PAYLOAD | SIGN),
last_notif : bstr(0x45 | OPT | 0xff | PAYLOAD | SIGN),
join_uri : "coap://myGM/ace-group/myGroup",
sec_gp : "myGroup"
}
}
C_2
[ Unicast w/ OSCORE ]
S
/r
0.05 (FETCH)
Token: 0x01
OSCORE: {kid: 2 ; piv: 201 ; ...}
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (Register),
<Other class E options>
}
(S increments the observer counter for the group observation of /r .)

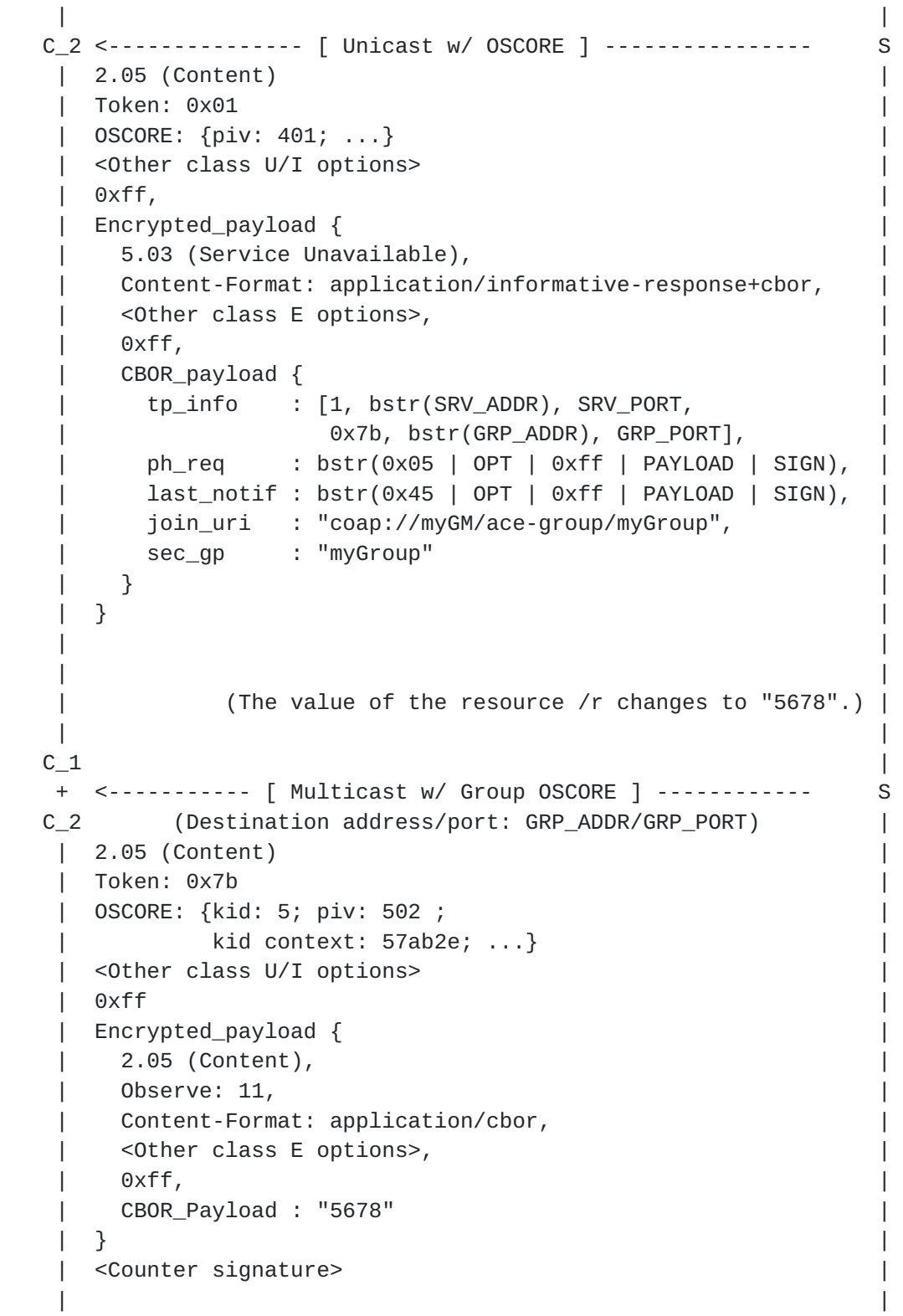



Figure 7: Example of group observation with Group OSCORE

The two external_aad used to encrypt and countersign the multicast notification above have 'request_kid' = 5, 'request_piv' = 501 and 'request_kid_context' = 0x57ab2e. These values are specified in the 'kid', 'piv' and 'kid context' field of the OSCORE option of the phantom observation request, which is encoded in the 'ph_req' parameter of the unicast informative response to the two clients. Thus, the two clients can build the two same external_aad for decrypting and verifying this multicast notification and the following ones.

## 9. Intermediaries

This section specifies how the approach presented in Section 2 and Section 3 works when a proxy is used between the clients and the server. In addition to what 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.

Upon receiving an informative response, the proxy performs as specified for the client in Section 3, with the peculiarity that "consuming" the last notification (if present) means populating its 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 will listen to the IP multicast address and port number indicated by the server in the informative response, as 'cli_addr' and 'cli_port' element of 'req_info' under the 'tp_info' parameter, respectively (see Section 2.2.1.1). Furthermore, multicast notifications will match the phantom request stored at the proxy, based on the Token value specified in the 'token' element of 'req_info' under the 'tp_info' parameter in the informative response.

Then, the proxy performs the following actions.

* If the 'last_notif' field is not present, the proxy responds to the client with an Empty Acknowledgement (if indicated by the message type, and if it has not already done so).

* If the 'last_notif' field is present, the proxy rebuilds the latest multicast notification, as defined in Section 3. Then, the proxy responds 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 to 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 responds to the client either with the latest multicast notification if available from its cache, or with an Empty Acknowledgement otherwise, as defined above.

An example is provided in Appendix E.

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 listens to an IP multicast address 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.

## 10. Intermediaries Together with End-to-End Security

As defined in Section 7, Group OSCORE can be used to protect multicast notifications end-to-end between the origin server and the clients. In such a case, additional actions are required when also the informative responses from the origin server are protected specifically end-to-end, by using OSCORE or Group OSCORE.

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.

Then, differently from what defined in Section 9, each proxy receiving an informative response simply forwards it back to the client that has sent the corresponding observation request. Note that the proxy does not even realize the message to be an actual informative response, since the outer Code field is set to 2.05 (Content).

Upon receiving the informative response, the client does not configure an observation of the target resource. Instead, the 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 client includes the new Listen-To-Multicast-Responses CoAP option defined in Section 10.1, to provide that proxy with the transport-specific information required for receiving multicast notifications for the group observation.

Details on the additional message exchange and processing are defined in Section 10.2.

### 10.1. The Listen-To-Multicast-Responses Option

To allow the proxy to listen to the multicast notifications sent by the server, a new CoAP option is introduced. This option MUST be supported by clients 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-Responses and is intended only for requests. As summarized in Figure 8, the option is critical and proxy-unsafe.

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

C = Critical, U = Unsafe, N = NoCacheKey, R = Repeatable
(*) See below.

Figure 8: Listen-To-Multicast-Responses

The Listen-To-Multicast-Responses option includes the serialization of a CBOR array. This specifies transport-specific message information required for listening to the multicast notifications of a group observation, and 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 2.2.1 for the 'tp_info' parameter of the informative response (see Section 2.2).

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

## 10.2. Message Processing

Compared to Section 9, the following additions apply when informative responses are protected end-to-end between the origin server and the 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 in Section 7.3.1:

* The client performs all the additional decryption and verification steps of Section 7.3.1 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 B of [I-D.amsuess-core-cachable-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 Code field, the outer CoAP options and the encrypted payload (containing inner options, AEAD tag etc.) are the same 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 will usually be set in the phantom request already.

- The outer options Proxy-Scheme, Uri-Host and Uri-Port are included, and set to the same values they had in the original registration request sent by the client.

- The new option Listen-To-Multicast-Responses is included as an outer option. The value is set to the 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 to 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 counter signature.

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 Proxy-Scheme, Uri-Host and Uri-Port options from the ticket request.

* The proxy removes the Listen-To-Multicast-Responses option from the ticket request, and extracts the conveyed transport-specific information.

* The proxy rebuilds the phantom request associated to 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 counter signature.

* 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 will listen to the IP multicast address and port number indicated in the Listen-To-Multicast-Responses option, as 'cli_addr' and 'cli_port' element of the serialized CBOR array, respectively. Furthermore, multicast notifications will match the phantom request stored at the proxy, based on the Token value specified in the 'token' element of the serialized CBOR array in the Listen-To-Multicast-Responses option.

An example is provided in Appendix F.

## 11. Informative Response Parameters

This specification defines a number of fields used in the informative response message defined in Section 2.2.

The table below summarizes them and specifies the CBOR key to use instead of the full descriptive name. Note that the media type application/informative-response+cbor MUST be used when these fields are transported.

| Name | CBOR Key | CBOR Type | Reference |
|---|---|---|---|
| tp_info | 1 | array | Section 2.2 |
| ph_req | 2 | byte string | Section 2.2 |
| last_notif | 3 | byte string | Section 2.2 |
| join_uri | 4 | text string | Section 7.1 |
| sec_gp | 5 | text string | Section 7.1 |
| as_uri | 6 | text string | Section 7.1 |
| cs_alg | 7 | int / text string | Section 7.1 |
| cs_crv | 8 | int / text string | Section 7.1 |
| cs_kty | 9 | int / text string | Section 7.1 |
| cs_kenc | 10 | int | Section 7.1 |
| alg | 11 | int / text string | Section 7.1 |
| hkdf | 12 | int / text string | Section 7.1 |
| gp_material | 13 | map | Appendix C |
| srv_pub_key | 14 | byte string | Appendix C |
| srv_identifier | 15 | byte string | Appendix C |
| exp | 16 | uint | Appendix C |

Table 1

## 12. Transport Protocol Information

This specification defines some values of transport protocol identifiers to use within the 'tp_info' parameter of the informative response message defined in Section 2.2 of this specification.

According to the encoding specified in Section 2.2.1, these values are used for the 'tp_id' element of 'srv_addr', under the 'tp_info' parameter.

The table below summarizes them, specifies the integer value to use instead of the full descriptive name, and provides the corresponding full set of information elements to include in the 'tp_info' parameter.

| Transport Protocol | Description | Value | Srv Addr | Req Info | Reference |
|---|---|---|---|---|---|
| Reserved | This value is reserved | 0 | | | [This document] |
| UDP | UDP is used as per RFC7252 | 1 | tp_id<br>srv_host<br>srv_port | token<br>cli_host<br>?cli_port | [This document] |

Figure 9: Transport protocol information

## 13. Security Considerations

The same security considerations from [RFC7252][RFC7641][I-D.ietf-core-groupcomm-bis][RFC8613][I-D.ietf-core-oscore-groupcomm] hold for this document.

If multicast notifications are protected using Group OSCORE, the original registration requests and related unicast (notification) responses MUST also be secured, including and especially the informative responses from the server. This prevents on-path active adversaries from altering the conveyed IP multicast address and serialized phantom registration request. Thus, it ensures secure binding between every multicast notification for a same observed resource and the phantom registration request that started the group observation of that resource.

To this end, clients and servers SHOULD use OSCORE or Group OSCORE, so ensuring that the secure binding above is enforced end-to-end between the server and each observing client.

### 13.1. Listen-To-Multicast-Responses Option

The CoAP option Listen-To-Multicast-Responses defined in Section 10.1 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 10.2), 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 matching to 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 to incorrectly configure a group observation (e.g., by indicating a wrong multicast IP address) hence preventing the correct reception of multicast notifications and their forwarding to the clients; or to configure bogus group observations that are currently not active on the origin server.

In order to prevent what described above, the ticket requests conveying the Listen-To-Multicast-Responses option can be additionally protected hop-by-hop.

## 14. IANA Considerations

This document has the following actions for IANA.

### 14.1. Media Type Registrations

This specification registers the media type 'application/informative-response+cbor' for error messages as informative response defined in Section 2.2 of this specification, when carrying parameters encoded in CBOR. This registration follows the procedures specified in [RFC6838].

* Type name: application

* Subtype name: informative-response+cbor

* Required parameters: N/A

* Optional parameters: N/A

* Encoding considerations: Must be encoded as a CBOR map containing the parameters defined in Section 2.2 of [this document].

* Security considerations: See Section 13 of [this document].

* Interoperability considerations: N/A

* Published specification: [this document]

* Applications that use this media type: The type is used by CoAP servers and clients that support error messages as informative response defined in Section 2.2 of [this document].

* Fragment identifier considerations: N/A

* Additional information: N/A

* Person & email address to contact for further information: iesg@ietf.org (mailto:iesg@ietf.org)

* Intended usage: COMMON

* Restrictions on usage: None

* Author: Marco Tiloca marco.tiloca@ri.se (mailto:marco.tiloca@ri.se)

* Change controller: IESG

### 14.2. CoAP Content-Formats Registry

IANA is asked to add the following entry to the "CoAP Content-Formats" sub-registry defined in Section 12.3 of [RFC7252], within the "Constrained RESTful Environments (CoRE) Parameters" registry.

Media Type: application/informative-response+cbor

Encoding: -

ID: TBD

Reference: [this document]

### 14.3. Informative Response Parameters Registry

This specification establishes the "Informative Response Parameters" IANA registry. The registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 14.6.

The columns of this registry are:

* Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

* CBOR Key: This is the value used as CBOR key of the item. These values MUST be unique. The value can be a positive integer, a negative integer, or a string.

* CBOR Type: This contains the CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.

* Reference: This contains a pointer to the public specification for the item.

This registry has been initially populated by the values in Section 11. The "Reference" column for all of these entries refers to sections of this document.

### 14.4. CoAP Transport Information Registry

This specification defines the subregistry "CoAP Transport Information" within the "CoRE Parameters" registry. The registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 14.6.

The columns of this registry are:

* Transport Protocol: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

* Description: Text giving an overview of the transport protocol referred by this item.

* Value: CBOR abbreviation for the transport protocol referred by this item. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

* Server Addr: List of elements providing addressing information of the server.

* Req Info: List of elements providing transport-specific information related to a pertinent CoAP request. Optional elements are prepended by '?'.

* Reference: This contains a pointer to the public specification for the item.

This registry has been initially populated by the values in Section 12. The "Reference" column for all of these entries refers to sections of this document.

### 14.5. 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 |
|---|---|---|
| TBD | Multicast-Response-Feedback-Divider | [This document] |
| TBD | Listen-To-Multicast-Responses | [This document] |

### 14.6. Expert Review Instructions

The IANA registries established in this document are defined as expert review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments, code points in other ranges should not be assigned for testing.

* Specifications are required for the standards track range of point assignment. Specifications should exist for specification required ranges, but early assignment before a specification is available is considered to be permissible. Specifications are needed for the first-come, first-serve range if they are expected to be used outside of closed environments in an interoperable way.

When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

* Experts should take into account the expected usage of fields when approving point assignment. The fact that there is a range for standards track documents does not mean that a standards track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

# 15. References

## 15.1. Normative References

[COSE.Algorithms]
IANA, "COSE Algorithms", <https://www.iana.org/assignments/cose/cose.xhtml#algorithms>.

[COSE.Elliptic.Curves]
IANA, "COSE Elliptic Curves", <https://www.iana.org/assignments/cose/cose.xhtml#elliptic-curves>.

[COSE.Key.Types]
IANA, "COSE Key Types", <https://www.iana.org/assignments/cose/cose.xhtml#key-type>.

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-09, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-ace-key-groupcomm-oscore-09.txt>.

[I-D.ietf-ace-oscore-profile]
Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-profile-15, 26 January 2021, <http://www.ietf.org/internet-drafts/draft-ietf-ace-oscore-profile-15.txt>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-02, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-core-groupcomm-bis-02.txt>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-10, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-core-oscore-groupcomm-10.txt>.

[I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020, <http://www.ietf.org/internet-drafts/draft-ietf-cose-rfc8152bis-algs-12.txt>.

[I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-14, 24 September 2020, <http://www.ietf.org/internet-drafts/draft-ietf-cose-rfc8152bis-struct-14.txt>.

[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>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

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

[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>.

[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>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

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

[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>.

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

[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>.

[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/info/rfc8949>.

### 15.2. Informative References

[I-D.amsuess-core-cachable-oscore]
Amsuess, C. and M. Tiloca, "Cachable OSCORE", Work in Progress, Internet-Draft, draft-amsuess-core-cachable-oscore-00, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-amsuess-core-cachable-oscore-00.txt>.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for

Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-authz-36, 16 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-ace-oauth-authz-36.txt>.

[I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-09, 30 September 2019, <http://www.ietf.org/internet-drafts/draft-ietf-core-coap-pubsub-09.txt>.

[I-D.ietf-core-coral]
Hartke, K., "The Constrained RESTful Application Language (CoRAL)", Work in Progress, Internet-Draft, draft-ietf-core-coral-03, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-core-coral-03.txt>.

[I-D.ietf-core-resource-directory]
Amsuess, C., Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE Resource Directory", Work in Progress, Internet-Draft, draft-ietf-core-resource-directory-26, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-core-resource-directory-26.txt>.

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-40, 20 January 2021, <http://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-40.txt>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-07, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-tiloca-core-oscore-discovery-07.txt>.

[MOBICOM99] Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S., and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network", Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking , August 1999, <https://people.eecs.berkeley.edu/~culler/cs294-f03/papers/bcast-storm.pdf>.

[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>.

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

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/info/rfc8747>.

## Appendix A. Different Sources for Group Observation Data

While the clients usually receive the phantom registration request and other information related to the group observation through an Informative Response, the same data can be made available through different services, such as the following ones.

### A.1. Topic Discovery in Publish-Subscribe Settings

In a Publish-Subscribe scenario ([I-D.ietf-core-coap-pubsub]), a group observation can be discovered along with topic metadata. For instance, a discovery step can make the following metadata available.

This example assumes a CoRAL namespace [I-D.ietf-core-coral], that contains properties analogous to those in the content-format application/informative-response+cbor.

Request:

```
GET </ps/topics?rt=oic.r.temperature>
Accept: CoRAL
```

Response:

```
2.05 Content
Content-Format: CoRAL

rdf:type <http://example.org/pubsub/topic-list>
topic </ps/topics/1234> {
    tp_info [1, h"7b", h"20010db80100..0001", 5683,
             h"ff35003020010db8..1234", 5683],
    ph_req h"0160..",
    last_notif h"256105.."
}
```

Figure 10: Group observation discovery in a Pub-Sub scenario

With this information from the topic discovery step, the client can already set up its multicast address and start receiving multicast notifications.

In heavily asymmetric networks like municipal notification services, discovery and notifications do not necessarily need to use the same network link. For example, a departure monitor could use its (costly and usually-off) cellular uplink to discover the topics it needs to update its display to, and then listen on a LoRA-WAN interface for receiving the actual multicast notifications.

## A.2. Introspection at the Multicast Notification Sender

For network debugging purposes, it can be useful to query a server that sends multicast responses as matching a phantom registration request.

Such an interface is left for other documents to specify on demand. As an example, a possible interface can be as follows, and rely on the already known Token value of intercepted multicast notifications, associated to a phantom registration request.

Request:

```
GET </.well-known/core/mc-sender?token=6464>
```

Response:

```
2.05 Content
Content-Format: application/informative-response+cbor

{
    'tp_info': [1, h"7b", h"20010db80100..0001", 5683,
                h"ff35003020010db8..1234", 5683],
    'ph_req': h"0160..",
    'last_notif' : h"256105.."
}
```

Figure 11: Group observation discovery with server introspection

For example, a network sniffer could offer sending such a request when unknown multicast notifications are seen on a network. Consequently, it can associate those notifications with a URI, or decrypt them, if member of the correct OSCORE group.

## Appendix B. Pseudo-Code for Rough Counting of Clients

This appendix provides a description in pseudo-code of the two algorithms used for the rough counting of active observers, as defined in Section 6.

In particular, Appendix B.1 describes the algorithm for the client side, while Appendix B.2 describes an optimized version for constrained clients. Finally, Appendix B.3 describes the algorithm for the server side.

### B.1. Client Side

```
   input:  int Q, // Value of the MRFD option
           int LEISURE_TIME, // DEFAULT_LEISURE from RFC 7252,
                             // unless overridden

   output: None


   int RAND_MIN = 0;
   int RAND_MAX = (2**Q) - 1;
   int I = randomInteger(RAND_MIN, RAND_MAX);

   if (I == 0) {
       float fraction = randomFloat(0, 1);

       Timer t = new Timer();
       t.setAndStart(fraction * LEISURE_TIME);
       while(!t.isExpired());

       Request req = new Request();
       // Initialize as NON and with maximum
       // No-Response settings, set options ...

       Option opt = new Option(OBSERVE);
       opt.set(0);
       req.setOption(opt);

       opt = new Option(MRFD);
       req.setOption(opt);

       req.send(SRV_ADDR, SRV_PORT);
   }
```

### B.2. Client Side - Optimized Version

```
   input:  int Q, // Value of the MRFD option
           int LEISURE_TIME, // DEFAULT_LEISURE from RFC 7252,
                             // unless overridden

   output: None


   const unsigned int UINT_BIT = CHAR_BIT * sizeof(unsigned int);

   if (respond_to(Q) == true) {
       float fraction = randomFloat(0, 1);

       Timer t = new Timer();
       t.setAndStart(fraction * LEISURE_TIME);
       while(!t.isExpired());

       Request req = new Request();
       // Initialize as NON and with maximum
       // No-Response settings, set options ...

       Option opt = new Option(OBSERVE);
       opt.set(0);
       req.setOption(opt);

       opt = new Option(MRFD);
       req.setOption(opt);

       req.send(SRV_ADDR, SRV_PORT);
   }

   bool respond_to(int Q) {
       while (Q >= UINT_BIT) {
           if (rand() != 0) return false;
           Q -= UINT_BIT;
       }
       unsigned int mask = ~((~0u) << Q);
       unsigned int masked = mask & rand();
       return masked == 0;
   }
```

### B.3. Server Side

```
   input:  int COUNT, // Current observer counter
           int M, // Desired number of confirmations
           int MAX_CONFIRMATION_WAIT,
           Response notification, // Multicast notification to send

   output: int NEW_COUNT // Updated observer counter
```

```
int D = 4; // Dampener value
int RETRY_NEXT_THRESHOLD = 4;
float CANCEL_THRESHOLD = 0.2;

int N = max(COUNT, 1);
int Q = max(ceil(log2(N / M)), 0);
Option opt = new Option(MRFD);
opt.set(Q);

notification.setOption(opt);
<Finalize the notification message>
notification.send(GRP_ADDR, GRP_PORT);

Timer t = new Timer();
t.setAndStart(MAX_CONFIRMATION_WAIT); // Time t1
while(!t.isExpired());

// Time t2

int R = <number of requests to the target resource
         between t1 and t2, with the MRFD option>;

int E = R * (2**Q);

// Determine after how many multicast notifications
// the next count update will be performed
if ((R == 0) || (max(E/N, N/E) > RETRY_NEXT_THRESHOLD)) {
    <Next count update with the next multicast notification>
}
else {
    <Next count update after 10 multicast notifications>
}

// Compute the new count estimate
int COUNT_PRIME = <current value of the observer counter>;
int NEW_COUNT = COUNT_PRIME + ((E - N) / D);

// Determine whether to cancel the group observation
if (NEW_COUNT < CANCEL_THRESHOLD) {
    <Cancel the group observation>;
    return 0;
}

return NEW_COUNT;
```

## Appendix C. OSCORE Group Self-Managed by the Server

For simple settings, where no pre-arranged group with suitable memberships is available, the server can be responsible to setup and manage the OSCORE group used to protect the group observation.

In such a case, a client would implicitly request to join the OSCORE group when sending the observe registration request to the server. When replying, the server includes the group keying material and related information in the informative response (see Section 2.2).

Additionally to what defined in Section 2, the CBOR map in the informative response payload contains the following fields, whose CBOR labels are defined in Section 11.

* 'gp_material': this element is a CBOR map, which includes what the client needs in order to set up the Group OSCORE Security Context.

  This parameter has as value a subset of the Group_OSCORE_Input_Material object, which is defined in Section 6.4 of [I-D.ietf-ace-key-groupcomm-oscore] and extends the OSCORE_Input_Material object encoded in CBOR as defined in Section 3.2.1 of [I-D.ietf-ace-oscore-profile].

  In particular, the following elements of the Group_OSCORE_Input_Material object are included, using the same CBOR labels from the OSCORE Security Context Parameters Registry, as in Section 6.4 of [I-D.ietf-ace-key-groupcomm-oscore].

  - 'ms', 'contexId', 'cs_alg', 'cs_params', 'cs_key_params', 'cs_key_enc'. These elements MUST be included.

  - 'hkdf', 'alg', 'salt'. These elements MAY be included.

  The following elements of the Group_OSCORE_Input_Material object MUST NOT be included.

  - 'group_senderId', 'ecdh_alg', 'ecdh_params', 'ecdh_key_params'.

* 'srv_pub_key': this element is a CBOR byte string, which wraps the serialization of the public key that the server uses in the OSCORE group. If the public key of the server is encoded as a COSE_Key (see the 'cs_key_enc' element of the 'gp_material' parameter), it includes 'kid' specifying the Sender ID that the server has in the OSCORE group.

* 'srv_identifier': this element MUST be present only if 'srv_pub_key' is also present and, at the same time, the used encoding for the public key of the server does not allow to specify a Sender ID within the associated public key. Otherwise, it MUST NOT be present. If present, this element is a CBOR byte string, with value the Sender ID that the server has in the OSCORE group.

* 'exp': with value the expiration time of the keying material of the OSCORE group specified in the 'gp_material' parameter, encoded as a CBOR unsigned integer. This field contains a numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what specified for NumericDate in Section 2 of [RFC7519].

A client receiving an informative response uses the information above to set up the Group OSCORE Security Context, as described in Section 2 of [I-D.ietf-core-oscore-groupcomm]. Note that the client does not obtain a Sender ID of its own, hence it installs a Security Context that a "silent server" would, i.e. without Sender Context. From then on, the client uses the received keying material to process the incoming multicast notifications from the server.

The server complies with the following points.

* The server MUST NOT self-manage OSCORE groups and provide the related keying material in the informative response for any other purpose than the protection of group observations, as defined in this document.

  The server MAY use the same self-managed OSCORE group to protect the phantom request and the multicast notifications of multiple group observations it hosts.

* The server MUST NOT provide in the informative response the keying material of other OSCORE groups it is or has been a member of.

After the time indicated in the 'exp' field:

* The server MUST stop using the keying material and MUST cancel the group observations for which that keying material is used (see Section 2.5). If the server creates a new group observation as a replacement or follow-up using the same OSCORE group:

  - The server MUST update the Master Secret.

- The server MUST update the ID Context (Gid). Consistently with Section 2.3 of [I-D.ietf-core-oscore-groupcomm], the server MUST assign an ID Context that it has never assigned before in the OSCORE group.

- The server MAY update the Master Salt.

* The client MUST stop using the keying material and MAY re-register the observation at the server.

Before the key material has expired, the server can send a multicast response with response code 5.03 (Service Unavailable) to the observing clients, protected with the current key material. In particular, this is an informative response (see Section 2.2) and contains the abovementioned parameters for the next group keying material to be used. Alternatively, the server can simply cancel the group observation (see Section 2.5), which results in the eventual re-registration of the clients that are still interested in the group observation.

Applications requiring backward security and forward security are REQUIRED to use an actual group joining process (usually through a dedicated Group Manager), e.g. the ACE joining procedure defined in [I-D.ietf-ace-key-groupcomm-oscore]. The server can facilitate the clients by providing them information about the OSCORE group to join, as described in Section 7.1.

## Appendix D. Phantom Request as Deterministic Request

In some settings, the server can assume that all the approaching clients already have the exact phantom observation request to use.

For instance, the clients can be pre-configured with the phantom observation request, or they may be expected to retrieve it through dedicated means (see Appendix A), before sending an observe registration request to the server.

If Group OSCORE is used to protect the group observation (see Section 7), and the OSCORE group supports the concept of Deterministic Client [I-D.amsuess-core-cachable-oscore], then the server and each client in the OSCORE group can independently protect the phantom observation request possibly available as plain CoAP message. To this end, they use the approach defined in Section 2 of [I-D.amsuess-core-cachable-oscore] to compute a protected deterministic request, against which the protected multicast notifications will match for the group observation in question.

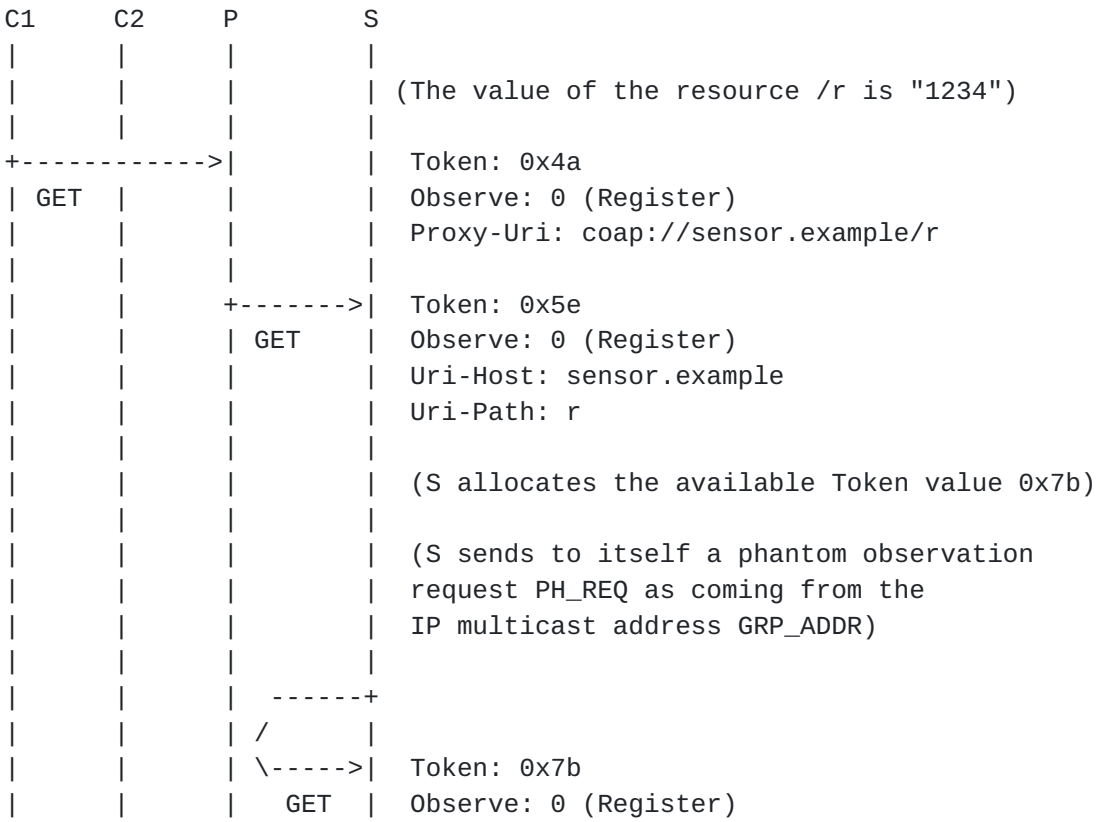
Note that, if the optimization defined in Appendix C is also used, the error informative response from the server has to include additional information, i.e. the Sender ID of the Deterministic Client in the OSCORE group, and the hash algorithm used to compute the deterministic request (see Section 2.3.1 of [I-D.amsuess-core-cachable-oscore]).

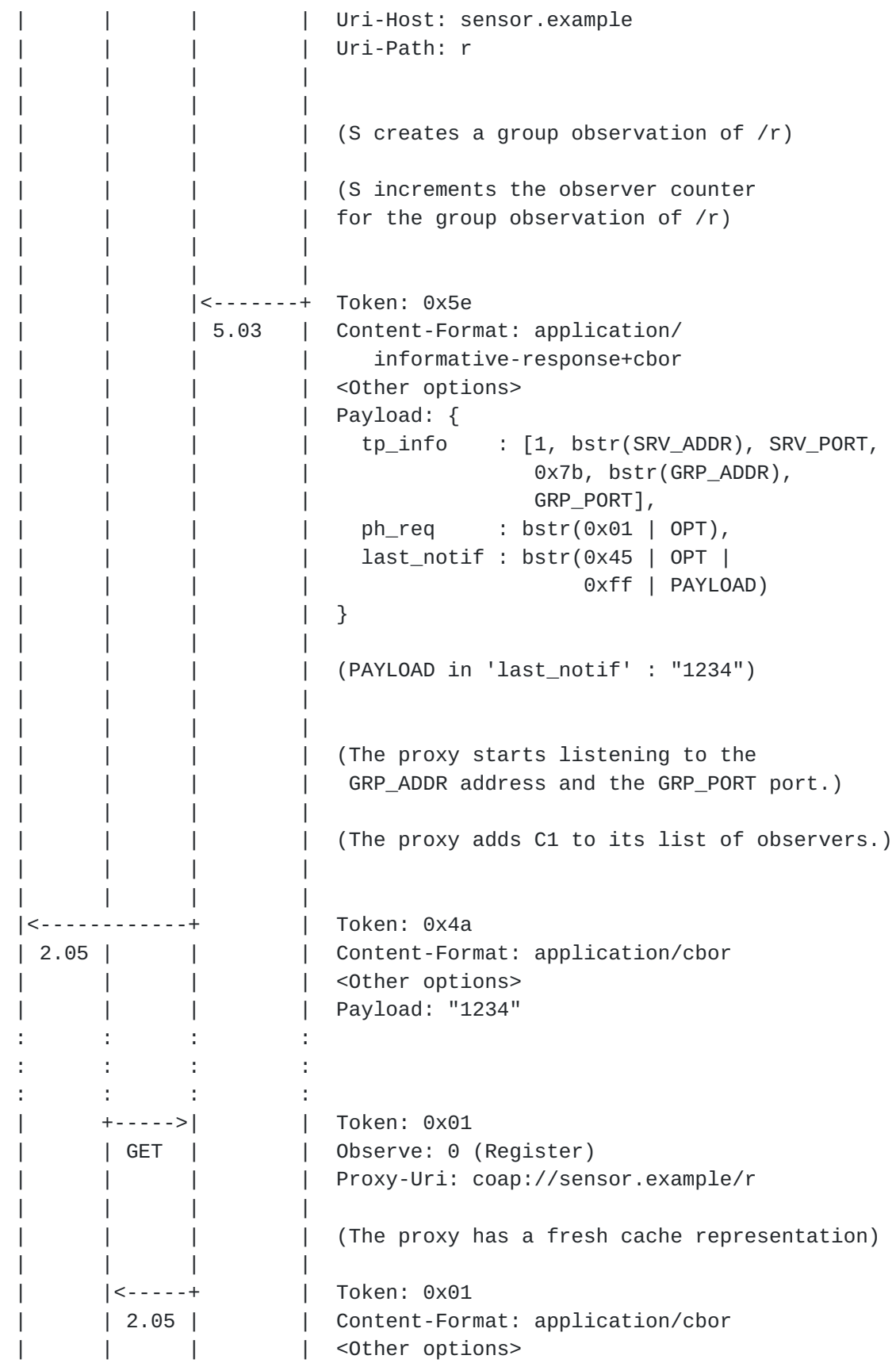
This optimization allows the server to not provide the same full phantom observation request to each client in the error informative response (see Section 2.2). That is, the informative response does not need to include the 'ph_req' parameter, but only the 'tp_info' parameter specifying the transport-specific information and (optionally) the 'last_notif' parameter specifying the latest sent multicast notification.

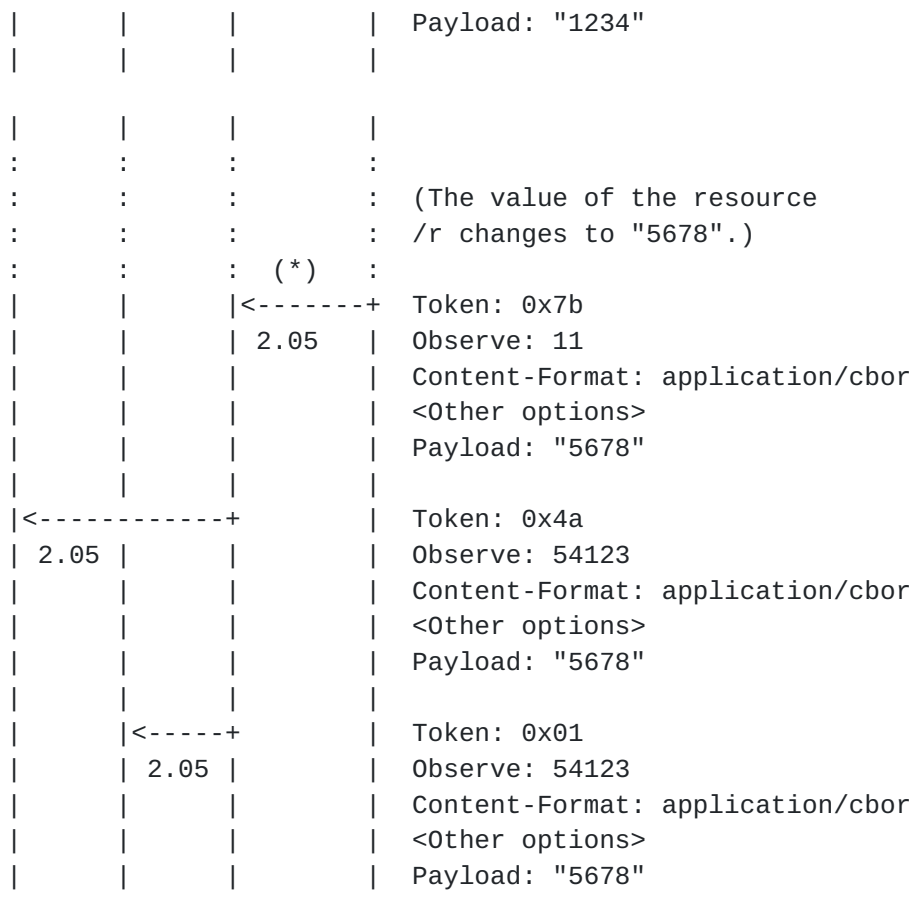
## Appendix E. Example with a Proxy

This section provides an example when a proxy P is used between the clients and the server. The same assumptions and notation used in Section 5 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.




Uri-Host: sensor.example
Uri-Path: r
(S creates a group observation of /r)
(S increments the observer counter
for the group observation of /r)
Token: 0x5e
5.03
Content-Format: application/
informative-response+cbor
<Other options>
Payload: {
tp_info : [1, bstr(SRV_ADDR), SRV_PORT,
0x7b, bstr(GRP_ADDR),
GRP_PORT],
ph_req : bstr(0x01 | OPT),
last_notif : 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
Content-Format: application/cbor
<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
Content-Format: application/cbor
<Other options>




(*) Sent over IP multicast to GROUP_ADDR:GROUP_PORT

Figure 12: 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.

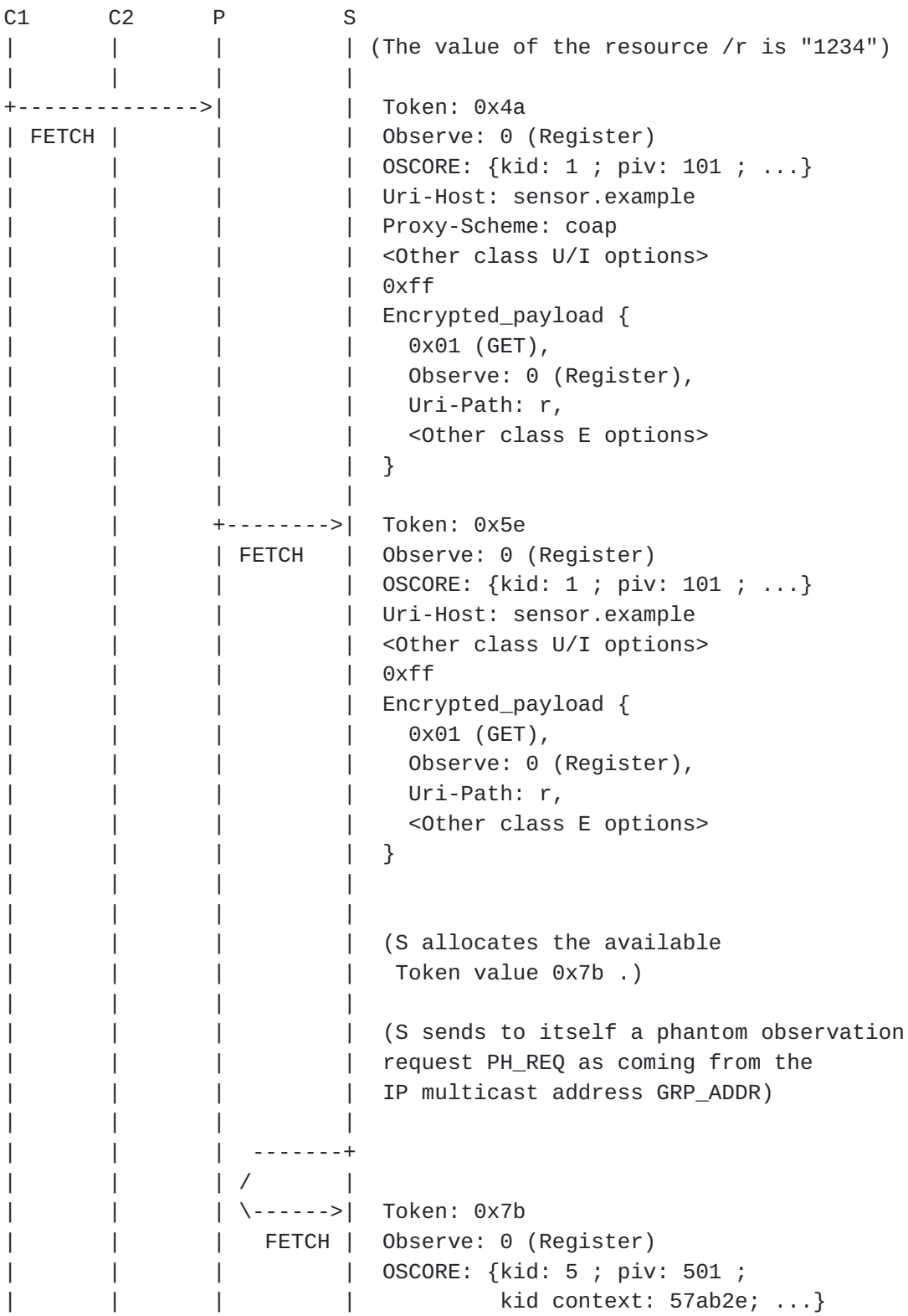
This behavior is mandated by Section 5 of [RFC7641], i.e. the proxy registers itself only once with the next hop and fans out the notifications it receives to all registered clients.

## Appendix F. Example with a Proxy and Group OSCORE

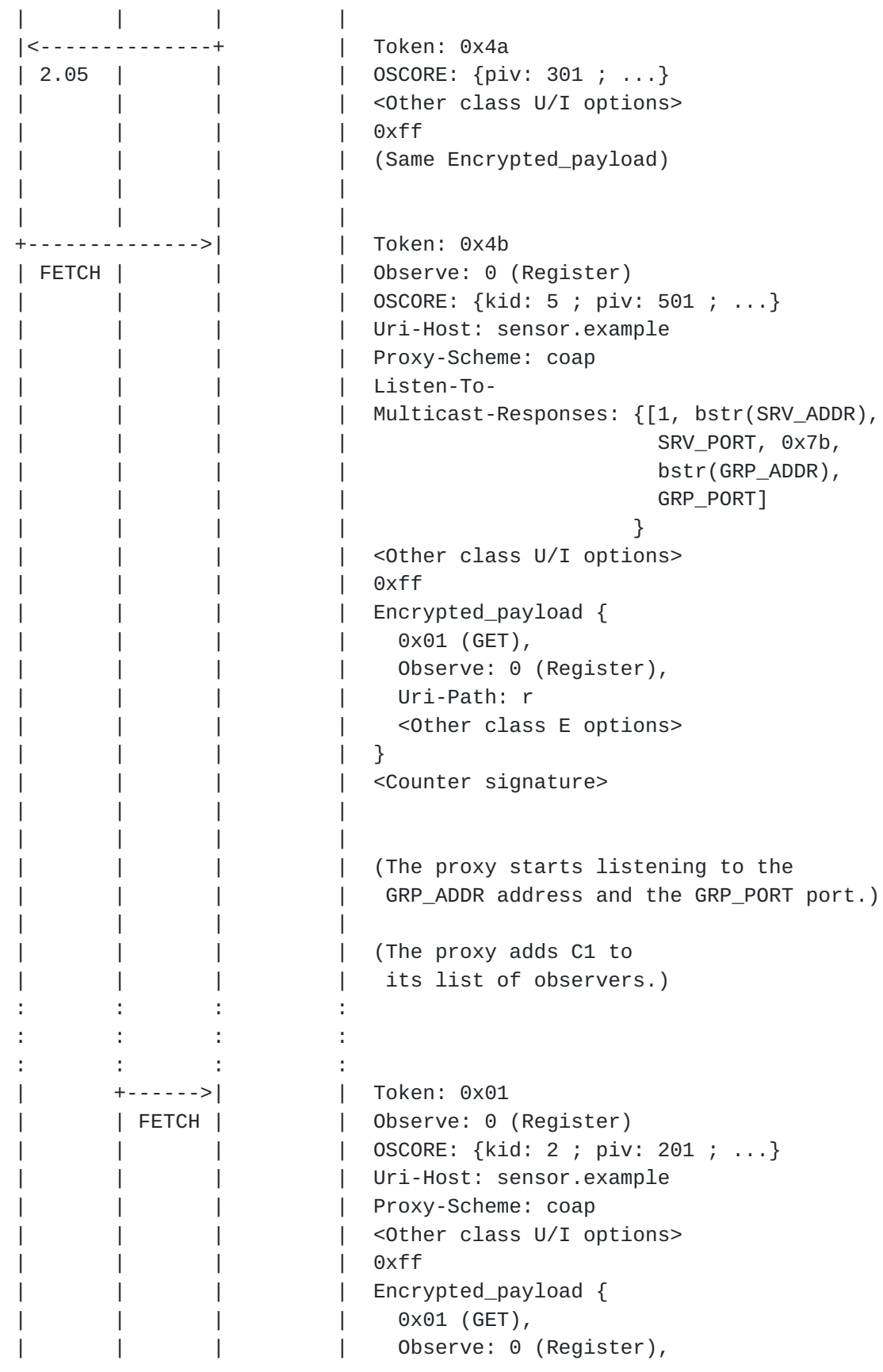
This section provides an example when a proxy P is used between the clients and the server, and Group OSCORE is used to protect multicast notifications end-to-end between the server and the clients.

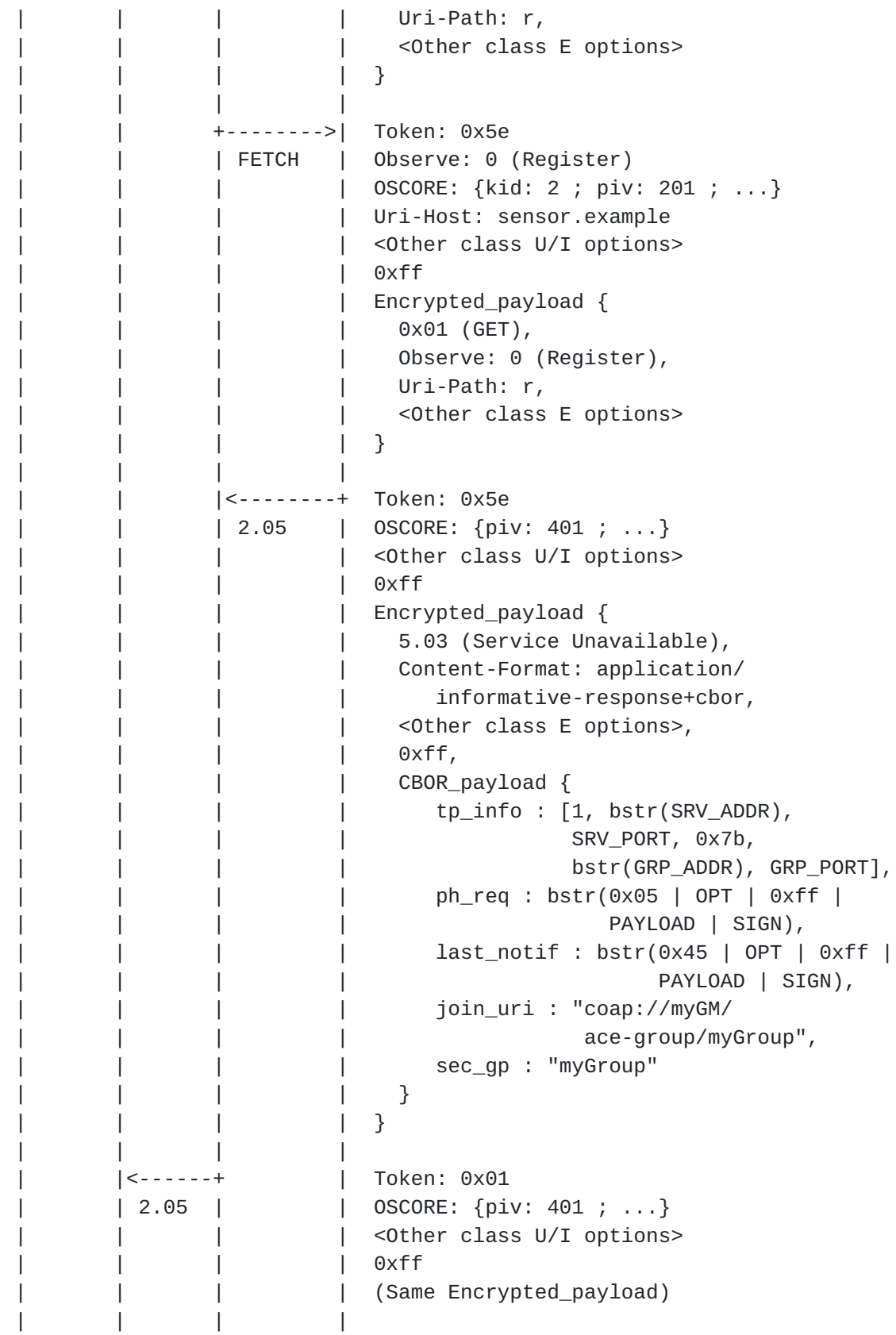
The same assumptions and notation used in Section 8 are used for this example. In addition, the proxy has address PRX_ADDR and listens to the port number PRX_PORT.

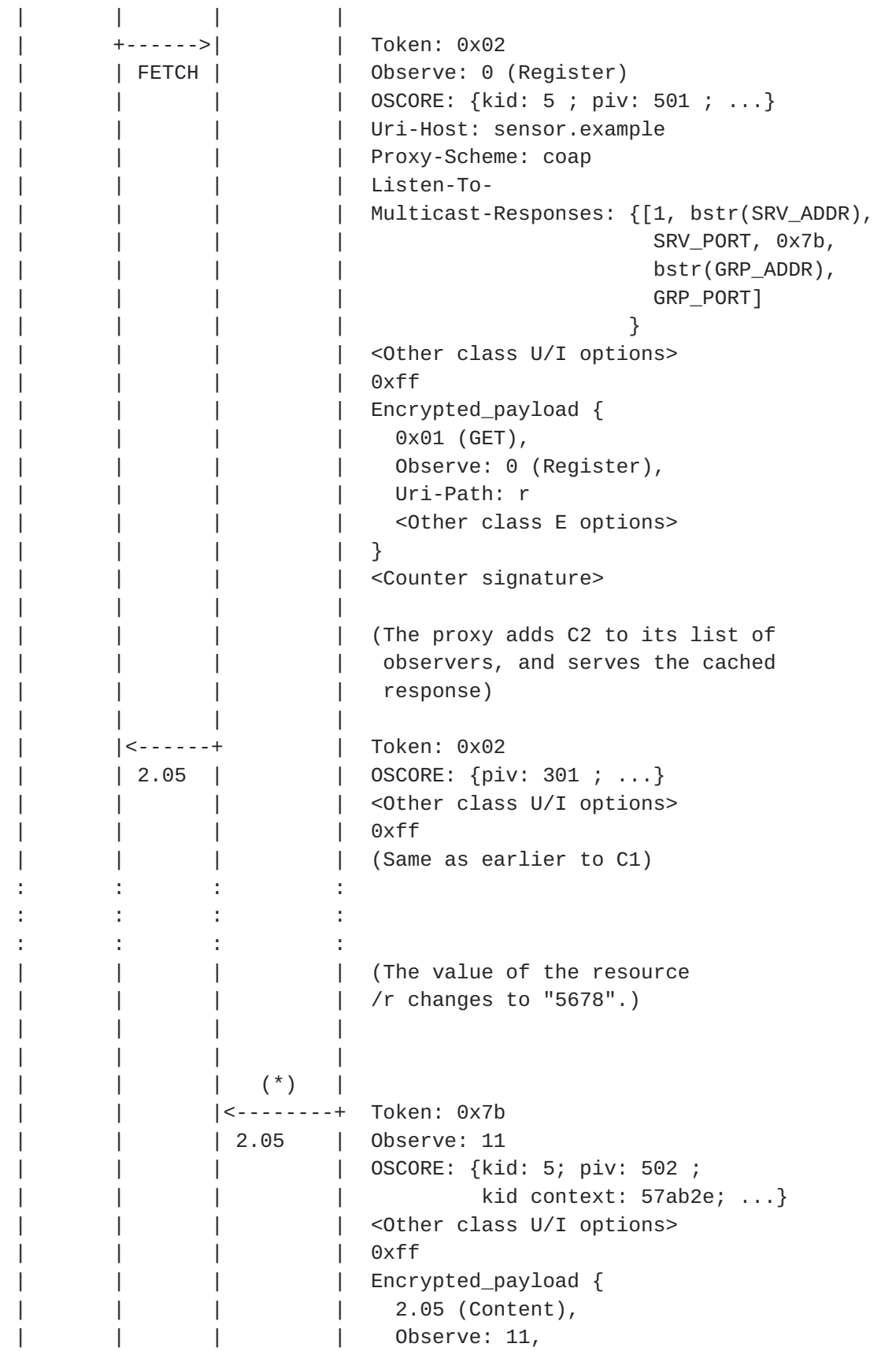
Unless explicitly indicated, all messages transmitted on the wire are sent over unicast and protected with OSCORE end-to-end between a client and the server.

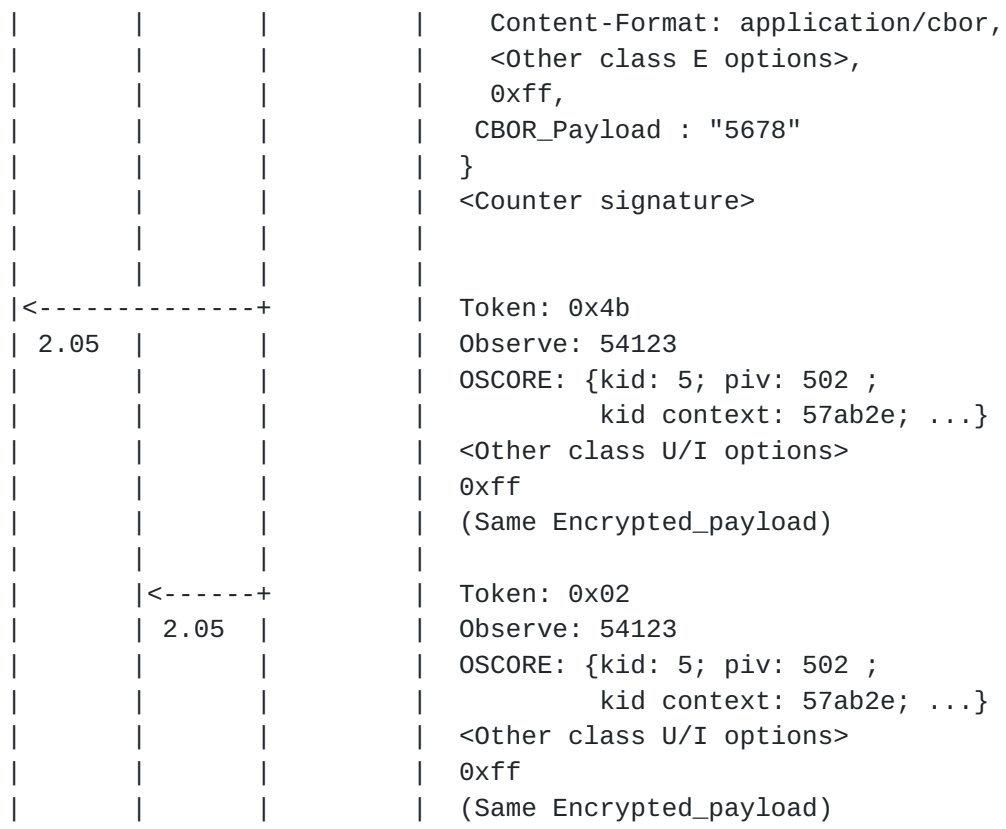


```
 |       |       |         |  Uri-Host: sensor.example
 |       |       |         |  <Other class U/I options>
 |       |       |         |  0xff
 |       |       |         |  Encrypted_payload {
 |       |       |         |    0x01 (GET),
 |       |       |         |    Observe: 0 (Register),
 |       |       |         |    Uri-Path: r
 |       |       |         |    <Other class E options>
 |       |       |         |  }
 |       |       |         |  <Counter signature>
 |       |       |         |
 |       |       |         |
 |       |       |         |  (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: {piv: 301 ; ...}
 |       |       |         |  <Other class U/I options>
 |       |       |         |  0xff
 |       |       |         |  Encrypted_payload {
 |       |       |         |    5.03 (Service Unavailable),
 |       |       |         |    Content-Format: application/
 |       |       |         |       informative-response+cbor,
 |       |       |         |    <Other class E options>,
 |       |       |         |    0xff,
 |       |       |         |    CBOR_payload {
 |       |       |         |       tp_info : [1, bstr(SRV_ADDR),
 |       |       |         |                   SRV_PORT, 0x7b,
 |       |       |         |                   bstr(GRP_ADDR), GRP_PORT],
 |       |       |         |       ph_req : bstr(0x05 | OPT | 0xff |
 |       |       |         |                      PAYLOAD | SIGN),
 |       |       |         |       last_notif : bstr(0x45 | OPT | 0xff |
 |       |       |         |                         PAYLOAD | SIGN),
 |       |       |         |       join_uri : "coap://myGM/
 |       |       |         |                    ace-group/myGroup",
 |       |       |         |       sec_gp : "myGroup"
 |       |       |         |    }
 |       |       |         |  }
 |       |       |         |
```


Token: 0x4a
OSCORE: {piv: 301 ; ...}
<Other class U/I options>
0xff
(Same Encrypted_payload)
2.05
Token: 0x4b
FETCH
Observe: 0 (Register)
OSCORE: {kid: 5 ; piv: 501 ; ...}
Uri-Host: sensor.example
Proxy-Scheme: coap
Listen-To-
Multicast-Responses: {[1, bstr(SRV_ADDR),
SRV_PORT, 0x7b,
bstr(GRP_ADDR),
GRP_PORT]
}
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (Register),
Uri-Path: r
<Other class E options>
}
<Counter signature>
(The proxy starts listening to the
GRP_ADDR address and the GRP_PORT port.)
(The proxy adds C1 to
its list of observers.)
Token: 0x01
FETCH
Observe: 0 (Register)
OSCORE: {kid: 2 ; piv: 201 ; ...}
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: 2 ; piv: 201 ; ...}
Uri-Host: sensor.example
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (Register),
Uri-Path: r,
<Other class E options>
}
Token: 0x5e
2.05
OSCORE: {piv: 401 ; ...}
<Other class U/I options>
0xff
Encrypted_payload {
5.03 (Service Unavailable),
Content-Format: application/
informative-response+cbor,
<Other class E options>,
0xff,
CBOR_payload {
tp_info : [1, bstr(SRV_ADDR),
SRV_PORT, 0x7b,
bstr(GRP_ADDR), GRP_PORT],
ph_req : bstr(0x05 | OPT | 0xff |
PAYLOAD | SIGN),
last_notif : bstr(0x45 | OPT | 0xff |
PAYLOAD | SIGN),
join_uri : "coap://myGM/
ace-group/myGroup",
sec_gp : "myGroup"
}
}
Token: 0x01
2.05
OSCORE: {piv: 401 ; ...}
<Other class U/I options>
0xff
(Same Encrypted_payload)


Token: 0x02
Observe: 0 (Register)
OSCORE: {kid: 5 ; piv: 501 ; ...}
Uri-Host: sensor.example
Proxy-Scheme: coap
Listen-To-
Multicast-Responses: {[1, bstr(SRV_ADDR),
SRV_PORT, 0x7b,
bstr(GRP_ADDR),
GRP_PORT]
}
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (Register),
Uri-Path: r
<Other class E options>
}
<Counter signature>
(The proxy adds C2 to its list of
observers, and serves the cached
response)
Token: 0x02
OSCORE: {piv: 301 ; ...}
<Other class U/I options>
0xff
(Same as earlier to C1)
(The value of the resource
/r changes to "5678".)
(*)
Token: 0x7b
Observe: 11
OSCORE: {kid: 5; piv: 502 ;
kid context: 57ab2e; ...}
<Other class U/I options>
0xff
Encrypted_payload {
2.05 (Content),
Observe: 11,




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

Figure 13: Example of group observation with a proxy and Group OSCORE

Unlike in the unprotected example in Appendix E, 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.

## Acknowledgments

The authors sincerely thank Carsten Bormann, Klaus Hartke, Jaime Jimenez, John Mattsson, Ludwig Seitz, 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
SE-16440 Stockholm Kista
Sweden

Email: marco.tiloca@ri.se

Rikard Hoeglund
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: rikard.hoglund@ri.se

Christian Amsuess
Hollandstr. 12/4
1020 Vienna
Austria

Email: christian@amsuess.com

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden

Email: francesca.palombini@ericsson.com