CoRE Working Group
Internet-Draft
Updates: 7252, 7641 (if approved)
Intended status: Standards Track
Expires: 24 October 2026

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

F. Palombini
Ericsson AB

22 April 2026

Observe Notifications as CoAP Multicast Responses
draft-ietf-core-observe-multicast-notifications-14

Abstract

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

Discussion Venues

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

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

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

Status of This Memo

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

This Internet-Draft will expire on 24 October 2026.

Copyright Notice

Table of Contents

1. Introduction

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

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

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 the same target resource to multiple observers as responses over IP multicast.

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

A different use case concerns clients observing the same registration resource at the CoRE Resource Directory [RFC9176]. For example, multiple clients can benefit from observation for discovering (to-be-created) groups that use the security protocol Group Object Security for Constrained RESTful Environments (Group OSCORE) [I-D.ietf-core-oscore-groupcomm], by retrieving from the Resource Directory updated links and descriptions to join those groups through the respective Group Manager [I-D.tiloca-core-oscore-discovery].

More in general, multicast notifications would be beneficial whenever several CoAP clients observe the same target resource at a CoAP server, and thus they could all be notified at once by means of a single response message. However, CoAP does not originally define response messages addressed to multiple clients, e.g., over IP multicast. This document 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 particular, the group of potential observers entrusts the server to manage the Token space for multicast notifications. Building on that, the server provides all the observers of a target resource with the same Token value to bind to their own observation, by sending a unicast informative response to each observer client. That Token value is then used in every multicast notification for the target resource under that observation.

Second, this document 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 unicast informative 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], Concise Data Definition Language (CDDL) [RFC8610], Concise Binary Object Representation (CBOR) [RFC8949], Object Security for Constrained RESTful Environments (OSCORE) [RFC8613], Group OSCORE [I-D.ietf-core-oscore-groupcomm], and Constrained Resource Identifiers (CRIs) [I-D.ietf-core-href].

This document additionally defines the following terminology.

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

* Group observation: a resource observation associated with 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 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. Prerequisites

In order to use multicast notifications as defined in this document, the following prerequisites have to be fulfilled.

* The server and the clients need to be on a network where multicast notifications can reach a sufficiently large portion of the clients.

  This document focuses on a network setup where the clients are capable of listening to multicast traffic and can directly receive multicast notifications.

  Alternative network setups may leverage intermediaries such as proxies, e.g., in order to accommodate clients that are not able to directly listen to multicast traffic. How the method specified in this document can be used in such setups is discussed in [I-D.ietf-core-multicast-notifications-proxy].

* The server needs to be provisioned with multicast addresses whose Token space is placed under its control. On general purpose networks, unmanaged multicast addresses such as "All CoAP Nodes" (see Section 12.8 of [RFC7252]) are not suitable for this purpose.

* The server and the clients need to agree out-of-band that multicast notifications may be used.

  This document does not describe a way for a client to influence the server’s decision to start group observations and thus to use multicast notifications. This is done on purpose.

That is, the method specified in this document is expected to be used in situations where sending individual notifications is not feasible, or is not preferred beyond a certain number of clients observing a target resource.

If applications arise where a negotiation between the clients and the server does make sense, those applications are welcome to specify additional means for clients to opt in for receiving multicast notifications.

3. High-Level Overview of Available Variants

The method defined in this document fundamentally enables a server to set up a group observation. This is associated with a phantom observation request that is generated by the server and to which the multicast notifications of the group observation are bound.

Consistent with the scope of this document, the following assumes a network setup where the clients participating in the group observation are capable of listening to multicast traffic. In such a setup, the clients directly receive multicast notifications from the server. Alternative network setups that rely on intermediaries such as proxies are discussed in [I-D.ietf-core-multicast-notifications-proxy].

While the server can provide the phantom request in question to the interested clients as they reach out for registering to the group observation, the server may alternatively distribute the phantom request in advance by alternative means (e.g., see Appendix A). Clients that have already retrieved the phantom request can immediately start listening to multicast notifications.

The rest of this section provides an overview of the available variants to enforce a group observation, which differ as to whether exchanged messages are protected end-to-end between the observer clients and the server.

* Variant without end-to-end security - Messages pertaining to the group observation are not protected. This basic case is defined in Section 4 and Section 5 from the server and the client side, respectively. An example is provided in Section 7.

* Variant with end-to-end security - Messages pertaining to the group observation are protected end-to-end between the clients and the server, by using the security protocol Group OSCORE [I-D.ietf-core-oscore-groupcomm]. This case is defined in Section 9. An example is provided in Section 10.

If the participating endpoints using Group OSCORE also support the concept of Deterministic Client [I-D.ietf-core-cacheable-oscore], then the possible early distribution of the phantom request can specifically make available its smaller, plain version. Consequently, all the clients are able to compute the same protected phantom request to use (see Appendix D).

4. 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 number of traditional observations has been established on the target resource, the server decides to make the corresponding observer 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. The server is expected to keep the counter up-to-date over time, for instance by using the method described in Section 8. This allows the server to possibly terminate a group observation if, at some point in time, not enough clients are estimated to be still active and interested.

4.1. Request

Assuming that the server 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 that have:

* 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 it was 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 request from the group of potential observer clients. In particular, from then on, the server MUST use T as its own local Token value associated with 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 the initial multicast notification for the target resource, in response to the phantom observation request. This message is formatted like other multicast notifications (see Section 4.3) and MUST include the current representation of the target resource as its 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 which the server deletes the message INIT_NOTIF.

4.2. 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 a Confirmable message sent as a separate response (see Section 5.2.2 of [RFC7252]), MUST NOT have link-local source or destination addresses, and MUST NOT provide link-local or site-local addresses in the transport-specific information specified in its payload (see below).

The Content-Format of the informative response is set to "application/informative-response+cbor", which is registered in Section 14.2. The payload of the informative response is a CBOR map, whose fields use the CBOR abbreviations that are defined in Section 11.

When using the method specified in this document, the CBOR map conveyed as the payload of the informative response includes the following parameters with the semantics defined below. Other specifications may define different uses of the informative response for providing alternative information that is relevant to other protocols and applications.

* 'tp_info', with value a CBOR array. This includes the transport-specific information required to correctly receive multicast notifications that are bound to the phantom observation request. Typically, this comprises the Token value associated with the group observation, as well as the source and destination addressing information of the related multicast notifications. The CBOR array is formatted as defined in Section 4.2.1. This parameter MUST be included.

* 'ph_req', with value the byte serialization of the transport-independent information of the phantom observation request (see Section 4.1), encoded as a CBOR byte string. The value of the CBOR byte string is formatted as defined in Section 4.2.2.

  This parameter MAY be omitted if the phantom request is, in terms of transport-independent information, identical to the registration request from the client. Otherwise, this parameter MUST be included.

Note that the registration request from the client may indeed differ from the phantom observation request in terms of transport-independent information, but still be acceptable for the server to register the client as taking part in the group observation.

* '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 4.2.2. This parameter MAY be included.

* 'next_not_before', with value the number of seconds that will minimally elapse before the server sends the next multicast notification for the group observation of the target resource, encoded as a CBOR unsigned integer. This parameter MAY be included.

  This information can help a new client to align itself with the server's timeline, especially in scenarios where multicast notifications are regularly sent. Also, it can help synchronizing different clients when orchestrating content distribution through multicast notifications.

* 'ending', with value the time when the group observation of the target resource is planned to be canceled, encoded as a CBOR integer or as a CBOR floating-point number. The value is the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what is specified for NumericDate in Section 2 of [RFC7519]. This parameter MAY be included.

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

```
informative_response_payload = {
   0 => array, ; 'tp_info' (transport-specific information)
 ? 1 => bstr,  ; 'ph_req' (transport-independent information)
 ? 2 => bstr,  ; 'last_notif' (transport-independent information)
 ? 3 => uint,  ; 'next_not_before'
 ? 4 => ~time  ; 'ending'
}
```

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 a Confirmable message sent as a separate response (see Section 5.2.2 of [RFC7252]).

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.

4.2.1. Transport-Specific Message Information

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

```
tp_info = [
    tpi_server: CRI-no-local, ; Addressing information of the server
  ? tpi_details               ; Further information about the request
]

tpi_details = (
  + elements ; The number, format, and encoding of the elements
             ; depend on the scheme-id and authority of the CRI
             ; specified as tpi_server
)

CRI-no-local = [
  scheme-id,
  authority
]

scheme-id = nint  ; -1 - scheme-number

authority = [?userinfo, host, ?port]
userinfo  = (false, text .feature "userinfo")
host      = (host-ip // host-name)
host-name = (*text) ; lowercase, NFC labels
host-ip   = (bytes .size 4 //
               (bytes .size 16, ?zone-id))
zone-id   = text
port      = 0..65535
```

Figure 2: General Format of 'tp_info'

The following holds for the two elements 'tpi_server' and 'tpi_details'.

* The 'tpi_server' element MUST be present and specifies:

- The transport protocol used to transport a CoAP response from the server, i.e., a multicast notification in this document; and

- The addressing information of the server, i.e., the source addressing information of the multicast notifications that are sent for the group observation.

  Such addressing information MUST be equal to the source addressing information of the informative response sent by the server (see Section 4.3).

This element specifies a CRI [I-D.ietf-core-href], of which both 'scheme' and 'authority' are given, while 'path', 'query', and 'fragment' are not given.

Consistent with Section 5.1.1 of [I-D.ietf-core-href], the CRI scheme is given as a negative integer 'scheme-id'. In particular, a 'scheme-id' with value ID denotes the CRI scheme that has CRI scheme number equal to (-1 - ID). The latter identifies the corresponding registered URI scheme, per the associated entry in the "Uniform Resource Identifier (URI) Schemes" registry defined in [RFC7595] and updated in Section 11.1 of [I-D.ietf-core-href].

The combination of URI scheme and 'authority' component determines the CoAP transport used to distribute multicast notifications for the group observation. Note that:

- If the 'authority' component specifies a host-ip, then the 'scheme-id' (hence the URI scheme) is sufficient to identify the transport.

- If the 'authority' component specifies a host-name, then the consumer of the CRI has to resolve the host-name and consider the result together with the 'scheme-id' (hence the URI scheme) in order to identify the transport. For instance, DNS resolution can be used (e.g., as defined in [RFC9953]).

The identified transport determines what elements are required in the 'tpi_details' element of the 'tp_info' array, as well as what information they convey, their encoding, and their semantics. Those elements are specified in the 'Transport Information Details' column of the "CoAP Transport Information" registry for the entry associated with the identified CoAP transport (see Section 14.6)

* The ’tpi_details’ element MAY be present and specifies transport-specific information related to a pertinent request message, i.e., the phantom observation request in this document.

  The exact format of ’tpi_details’ depends on the CoAP transport, which is identified according to the CRI conveyed by the ’tpi_server’ element, as described above.

  In the "CoAP Transport Information" registry defined in Section 14.6 of this document, the entry corresponding to the identified CoAP transport specifies the list of elements composing ’tpi_details’ for that transport, as indicated in the ’Transport Information Details’ column. Within ’tpi_details’, its elements MUST be ordered according to what is specified in the ’Transport Information Details’ column of the "CoAP Transport Information" registry.

Section 12 defines an entry to be registered in the "CoAP Transport Information" registry, for the transport "CoAP over UDP". When such a transport is used, i.e., CoAP responses are transported over UDP as per [RFC7252] and [I-D.ietf-core-groupcomm-bis], the full encoding of the ’tp_info’ CBOR array is as defined in Section 4.2.1.1.

If a future specification defines the use of CoAP multicast notifications transported over different transport protocols, then that specification must perform the following actions, unless those have been already performed for different reasons:

* Define the elements in ’tpi_details’, as to what information they convey, their encoding, and their semantics.

* Register an entry in the "CoAP Transport Information" registry defined in Section 14.6 of this document.

* Register an entry in the "Uniform Resource Identifier (URI) Schemes" registry defined in [RFC7595] and updated in Section 11.1 of [I-D.ietf-core-href], where the value in the ’CRI Scheme Number’ column is (-1 - ID). In particular, ID is the negative integer to be used as ’scheme-id’ for CRIs conveyed by the ’tpi_server’ element and by elements in ’tpi_details’.

4.2.1.1. 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 ’tp_info’ CBOR array as follows.

* In the 'tpi_server' element, the CRI has 'scheme-id' with value -1 ("coap"), while 'authority' conveys addressing information of the server, i.e., the source addressing information of the multicast notifications that are sent for the group observation.

  This information consists of the IP address SRV_ADDR (expressed as a literal or as a host-name to be resolved) and the port number SRV_PORT of the server hosting the target resource, from where the server will send multicast notifications for the target resource.

* The 'tpi_details' element MUST be present and in turn includes the following two elements:

  - 'tpi_client' is a CRI, with the same format of 'tpi_server' (see Section 4.2.1). In particular, the CRI has 'scheme-id' with value -1 ("coap"), while 'authority' conveys the destination addressing information of the multicast notifications that the server sends for the group observation.

    This information consists of the IP multicast address GRP_ADDR (expressed as a literal or as a host-name to be resolved) and the port number GRP_PORT, where the server will send multicast notifications for the target resource.

  - 'tpi_token' is a CBOR byte string, whose value is the Token value of the phantom observation request generated by the server (see Section 4.1). Note that the same Token value is used for the multicast notifications bound to that phantom observation request (see Section 4.3).

The CDDL notation in Figure 3 describes the format of the 'tp_info' CBOR array when CoAP is transported over UDP.

```
tp_info_coap_udp = [
  tpi_server: CRI-no-local, ; Source addressing information
                            ; of the multicast notifications
  tpi_client: CRI-no-local, ; Destination addressing information
                            ; of the multicast notifications
  tpi_token: bstr           ; Token value of the phantom request and
                            ; associated multicast notifications
]
```

Figure 3: Format of 'tp_info' with UDP as Transport Protocol

The CBOR diagnostic notation in Figure 4 provides an example of the 'tp_info' CBOR array when CoAP is transported over UDP.

In the example, SRV_ADDR is 2001:db8::ab, SRV_PORT is 5683 (omitted in the CRI of 'tpi_server' as it is the default port number when CoAP is transported over UDP), GRP_ADDR is ff35:30:2001:db8::23, and GRP_PORT is 61616.

```
[ / tp_info /
  [ / tpi_server /
    -1,        / scheme-id -- equivalent to "coap" /
    h'20010db8000000000000000000000ab'  / host-ip /
  ],
  [ / tpi_client /
    -1,        / scheme-id -- equivalent to "coap" /
    h'ff35003020010db80000000000000023', / host-ip /
    61616                                  / port /
  ],
  h'7b'                              / tpi_token /
]
```

Figure 4: Example of 'tp_info' with UDP as Transport Protocol

4.2.2. Transport-Independent Message Information

For both the parameters 'ph_req' and 'last_notif' in the informative response, the value of the CBOR 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.

4.3. 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 such multicast notification is formatted as follows.

* It MUST be a Non-confirmable message.

* 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.

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

  The Token value T is specified by an element of 'tpi_details' within the 'tp_info' parameter, in the informative response sent to the observer clients. In particular, when transporting CoAP over UDP, the Token value is specified by the element 'tpi_token' (see Section 4.2.1.1).

* It MUST be sent from the same IP address SRV_ADDR and port number SRV_PORT where the corresponding informative responses are sent from by the server (see Section 4.2). That is, redirection MUST NOT be used.

  Note that, in most cases, such SRV_ADDR and SRV_PORT are those to which original observation requests are sent to by clients (see Section 5.1), unless those requests are sent to a multicast address (see [I-D.ietf-core-groupcomm-bis]).

  The addressing information above is provided to the observer clients through the CRI specified by the element 'tpi_server' within the 'tp_info' parameter, in the informative response (see Section 4.2.1).

* It MUST be sent to the IP multicast address GRP_ADDR and port number GRP_PORT.

  The addressing information above is provided to the observer clients through the CRI specified by an element of 'tpi_details' within the 'tp_info' parameter, in the informative response. In particular, when transporting CoAP over UDP, the CRI is conveyed by the element 'tpi_client' (see Section 4.2.1.1).

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

4.4. Congestion Control

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

* The multicast notifications MUST be Non-confirmable messages.

* In constrained environments such as low-power, lossy networks (LLNs), the server SHOULD only support multicast notifications for resources whose representation is small in size. Following related guidelines from Section 3.6 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 when using IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) (see Section 4 of [RFC4944]).

* 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 3.6 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. Ultimately, it is up to the server administrator to explicitly configure the most appropriate IP multicast scope.

* 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]). Furthermore, a goal for an appropriate transmission rate of multicast notifications is 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 to the server.

4.5. Cancellation

At a certain point in time, the server might want to cancel a group observation of a target resource. For instance, the server realizes that no clients or not enough clients are interested in taking part in the group observation anymore. Section 8 defines a possible approach that the server can use to make an assessment in this respect. Another reason is that the group observation has reached its ending time, as originally scheduled by the server.

In order to cancel the group observation, the server sends a multicast response with response code 5.03 (Service Unavailable), signaling that the group observation has been terminated. The response has the same Token value T of the phantom registration request, it has no payload, and it does not include an Observe Option.

The server sends the response to the same multicast IP address GRP_ADDR and port number GRP_PORT used to send the multicast notifications related to the target resource. Finally, the server releases the memory and network resources allocated for the group observation, and it especially frees up the Token value T used at its CoAP endpoint.

5. Client-Side Requirements

5.1. 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 have link-local source or destination addresses. If the server is not configured to accept registrations on that target resource specifically for a group observation, this would still result in a positive notification response to the client as described in [RFC7641], in the case that the server is able and willing to add the client to the list of observers.

In a particular setup, the information typically specified in the 'tp_info' parameter of the informative response (see Section 4.2) can be pre-configured on the server and the clients. For example, the destination multicast address and port number where to send multicast notifications for a group observation, as well as the associated Token value to use, can be set aside for particular tasks (e.g., enforcing observations of a specific resource). Alternative mechanisms can rely on using some bytes from the hash of the observation request as the last bytes of the multicast address or as part of the Token value.

In such a particular setup, the client may also have an early knowledge of the phantom request, i.e., it will be possible for the server to safely omit the parameter 'ph_req' from the informative response to the observation request (see Section 4.2). In this case, the client can include a No-Response Option [RFC7967] with value 16 in its Observe registration request, which results in the server suppressing the informative response. As a consequence, the observation request only informs the server that there is one additional client interested to take part in the group observation.

While the considered client is able to simply set up its multicast address and start receiving multicast notifications for the group observation, sending an observation request as above allows the server to increment the observer counter. This helps the server to assess the current number of clients interested in the group observation over time (e.g., by using the method defined in Section 8), which in turn can play a role in deciding to cancel the group observation (see Section 4.5).

5.2. Informative Response

Upon receiving the informative response defined in Section 4.2, the client has to identify the CoAP transport used to distribute multicast notifications for the group observation.

To this end, the client relies on the element 'tpi_server' within the 'tp_info' parameter of the informative response (see Section 4.2.1).

In particular, the client considers the CRI conveyed by 'tpi_server' and identifies the CoAP transport, by assessing together the 'authority' component and the URI scheme determined from 'scheme-id' (see Section 4.2.1).

After that, the client parses the remainder of the 'tp_info' array, i.e., the information conveyed by 'tpi_details', according to what is specified in the 'Transport Information Details' column of the "CoAP Transport Information" registry for the entry associated with the identified CoAP transport (see Section 14.6).

Then, the client performs the following steps.

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 by the CRI conveyed by the element 'tpi_server' within the 'tp_info' parameter, in the informative response (see Section 4.2.1).

     If the port number is not present in the CRI, the client MUST use as SRV_PORT the default port number defined for the identified CoAP transport (e.g., the default port number is 5683 when the transport is CoAP over UDP).

   * As destination address and port number, the IP multicast address GRP_ADDR and port number GRP_PORT. These are specified by the CRI conveyed by a dedicated element of

'tpi_details' within the 'tp_info' parameter, in the informative response. In particular, when transporting CoAP over UDP, the CRI is conveyed by the element 'tpi_client' (see Section 4.2.1.1).

If the port number is not present in the CRI, the client MUST use as GRP_PORT the default port number defined for the identified CoAP transport (e.g., the default port number is 5683 when the transport is CoAP over UDP).

2. The client rebuilds the phantom registration request as follows.

   * The client uses the Token value T that is specified by a dedicated element of 'tpi_details' within the 'tp_info' parameter, in the informative response. In particular, when transporting CoAP over UDP, the Token value is specified by the element 'tpi_token' (see Section 4.2.1.1).

   * If the 'ph_req' parameter is not present in the informative response, the client uses the transport-independent information from its original Observe registration request.

   * If the 'ph_req' parameter is present in the informative response, the client uses the transport-independent information specified in the parameter.

3. If the informative response includes the parameter 'ph_req' and the transport-independent information specified therein differs from the one in the original Observe registration request, then the client checks whether a response to the rebuilt phantom request can, if available in a cache entry, be used to satisfy the original observation request. If this is not the case, the client SHOULD explicitly withdraw from the group observation.

4. The client stores the phantom registration request, as associated with 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 with that group observation, with respect to the server. The particular way to achieve this is implementation specific.

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

   * The same Token value T used at Step 2; and

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

6. If the informative response includes the parameter 'last_notif', the client processes the multicast notification rebuilt at Step 5 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.

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

In addition to 'tpi_server', further elements of the 'tp_info' array can convey a CRI. The client MUST treat any CRI within the 'tp_info' array as invalid, if the 'authority' component is a host-name such that, when resolved, its combination with the URI scheme indicates multiple transports (see Section 4.2.1). As a possible way to verify if that is the case, the client can rely on DNS resolution (e.g., as defined in [RFC9953]).

If any of the expected fields in the informative response are absent, malformed, or invalid, the client MAY try sending a new registration request to the server (see Section 5.1). If the client chooses not to, then 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.

5.3. Notifications

After having successfully processed the informative response as defined in Section 5.2, 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].

5.4. Cancellation

At a certain point in time, a client may no longer be 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 a Non-confirmable message, it is optional for the client to reject it with a Reset message (see Section 3.5 of [RFC7641]).

If the server has canceled a group observation as defined in Section 4.5, 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 4.5.

6. Web Linking

The possible use of multicast notifications in a group observation MAY be indicated by a target attribute "gp-obs" in a web link [RFC8288] to a resource, e.g., using a link-format document [RFC6690].

The "gp-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 "gp-obs" attribute and any present value MUST be ignored by the recipient. The "gp-obs" attribute MUST NOT appear more than once in a given link-value; occurrences after the first MUST be ignored by the recipient.

The example in Figure 5 shows a use of the "gp-obs" attribute: the client does resource discovery on a server and gets back a list of resources, one of which includes the "gp-obs" attribute indicating that the server might send multicast notifications for observations of that resource. The CoRE Link-Format notation from Section 5 of [RFC6690] is used.

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

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

Figure 5: The Web Link

7. Example

The following example refers to two clients C1 and C2 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. The server starts the group observation upon receiving a registration request from a first client that wishes to start a traditional observation on the resource /r.

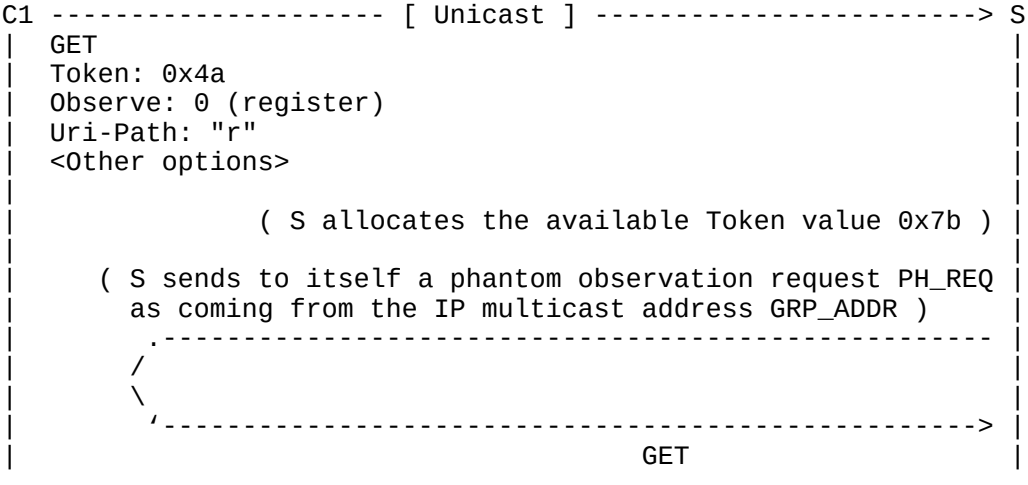
The following notation is used for the payload of the informative responses:

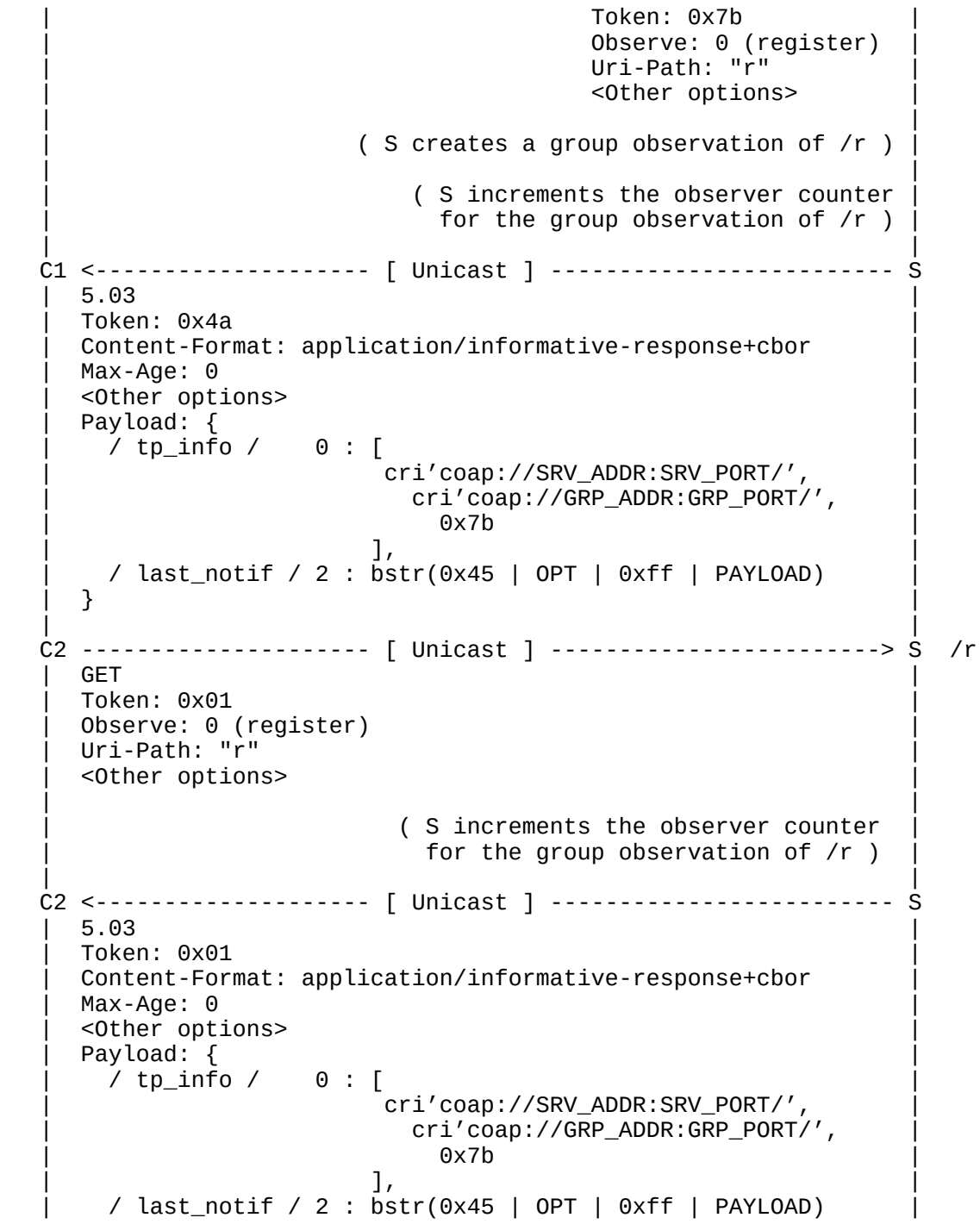
* The application-extension identifier "cri" defined in Section 3.4 of [I-D.ietf-cbor-edn-literals] is used to notate a CBOR Extended Diagnostic Notation (EDN) literal for a CRI.

* '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.




Token: 0x7b
Observe: 0 (register)
Uri-Path: "r"
<Other options>
(S creates a group observation of /r)
(S increments the observer counter
for the group observation of /r)
C1 <-- [Unicast] -- S
5.03
Token: 0x4a
Content-Format: application/informative-response+cbor
Max-Age: 0
<Other options>
Payload: {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b
],
/ last_notif / 2 : bstr(0x45 | OPT | 0xff | PAYLOAD)
}
C2 -- [Unicast] --> S /r
GET
Token: 0x01
Observe: 0 (register)
Uri-Path: "r"
<Other options>
(S increments the observer counter
for the group observation of /r)
C2 <-- [Unicast] -- S
5.03
Token: 0x01
Content-Format: application/informative-response+cbor
Max-Age: 0
<Other options>
Payload: {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b
],
/ last_notif / 2 : bstr(0x45 | OPT | 0xff | PAYLOAD)

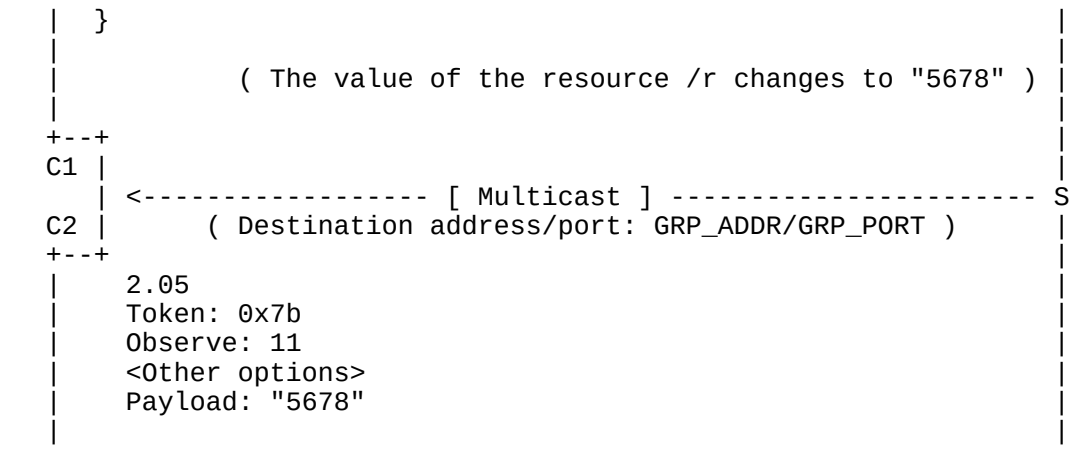


Figure 6: Example of Group Observation

8. Rough Counting of Clients in the Group Observation

This section specifies a method that the server can use to keep an estimate of still active and interested clients, without creating undue traffic on the network.

8.1. Feedback-Divider Option

This section defines the new CoAP option Feedback-Divider. By including the option in an outgoing message, a sender endpoint elicits a stochastic reaction and thereby a feedback from the message recipient(s) that would otherwise not react.

If the sender endpoint triggering a feedback is a client sending a request, the Feedback-Divider Option included in the request triggers the recipient server(s) to send a response with some probability. In particular, the Feedback-Divider Option stochastically overrides the possible response suppression performed by the server(s). Such an influence affects response suppression that can otherwise be performed, e.g., according to: default processing; the No-Response Option [RFC7967], if present in the request; an unmatching filter from the URI query component, when the request is sent over multicast and targets the /.well-known/core resource (see Section 4.1 of [RFC6690]).

If the sender endpoint triggering a feedback is a server sending a response, the feedback elicitation is currently limited to using observe notifications. In particular, the Feedback-Divider Option included in an observe notification triggers the recipient client(s) to send with some probability a new request to the server. Such a

request includes the Observe Option set to 0 (register) and is addressed to the same target resource for which the observe notification was sent. That is, a reacting client (re-)registers a regular unicast observation on the same target resource.

This document specifically defines how the Feedback-Divider Option is used when the sender endpoint triggering a feedback is a server sending multicast notifications. Note that it is not so useful to include the Feedback-Divider Option in an observe notification sent over unicast to a single client. That is, the server can more efficiently query a single client by means of an observe notification sent as a Confirmable message, thereby eliciting an Acknowledgement message in return.

The Feedback-Divider Option has the properties summarized in Table 1, which extends Table 4 of [RFC7252]. The option is not Critical, not Safe-to-Forward, and integer valued. Since the option is not Safe-to-Forward, the ’N’ column indicates a dash for "not applicable".

No.	C	U	N	R	Name	Format	Length	Default
TBD18	x	-			Feedback-Divider	uint	0-1	(none)

Table 1: The Feedback-Divider Option. C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Note to RFC Editor: In the table above, please replace TBD18 with the registered option number. Then, please delete this paragraph.

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

8.2. Processing on the Client Side

Upon receiving a response with a Feedback-Divider Option, a client that supports the option and is interested in continuing receiving multicast notifications for the target resource SHOULD acknowledge its interest, as described below.

The client picks an integer random number I, from 0 inclusive to Z = (2^Q) exclusive, where Q is the value specified in the option. If I is different from 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 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 of the source IP address and port number of the received multicast notification.

Since this Observe registration is only done for its side effect of looking as an attempted observation at the server, the client MUST send the unicast request as a Non-confirmable message and with the maximum No-Response setting [RFC7967]. In the request, the client MUST include a Feedback-Divider Option, whose value MUST be set to 0. As per Section 3.2 of [RFC7252], this is represented with an empty option value (a zero-length sequence of bytes). The client does not need to wait for responses and can keep processing further notifications on the same Token.

The client MUST ignore the Feedback-Divider Option, if the multicast notification is retrieved from the 'last_notif' parameter of an informative response (see Section 4.2). A client includes the 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.

Appendix B.1 and Appendix B.2 provide a description in pseudo-code of the operations above performed by the client.

8.3. 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 4), for instance by using the method described below.

8.3.1. Request for Feedback

When it wants to obtain a new estimated count, the server considers a number M of confirmations that 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 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 Feedback-Divider Option only in the first notification of such a burst.

8.3.2. Collection of Feedback

The server collects 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 4.2).

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 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 3.1.5 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.

8.3.3. Processing of Feedback

Once MAX_CONFIRMATION_WAIT seconds have passed, the server counts the R confirmations that have arrived as observation requests to the target resource, since the time when the latest multicast notification with the Feedback-Divider Option has been sent. In particular, the server considers an observation request as a confirmation from a client only if the request includes a Feedback-Divider Option with value 0.

Then, the server computes a feedback indicator as E = R * (2^Q). According to what is 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 considers 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 4.2).

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, which can be taken into account for possibly canceling the group observation (see Section 4.5).

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 a notification with Feedback-Divider with value Q = 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 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.

9. Protection of Multicast Notifications with Group OSCORE

A server can protect multicast notifications by using the security protocol Group OSCORE [I-D.ietf-core-oscore-groupcomm], thus ensuring that they are protected end-to-end for 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 already be a 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 9.1. Note that clients could already be members of the right OSCORE group, if they previously joined it to securely communicate with the same server or with other servers to access their resources.

Both the clients and the server MAY join the OSCORE group by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore] and based on the ACE framework for Authentication and Authorization in constrained environments [RFC9200]. When doing so, the server has to

join the group (also) with the roles of Requester and Responder. Instead, a client can join the group with any permitted role or combination of roles, unless it intends to send its original observation requests (see Section 5.1) protected with Group OSCORE. In such a case, the client has to join the group (also) as a Requester. Further details on how to discover the OSCORE group and join it are out of the scope of this document.

If multicast notifications are protected using Group OSCORE, then the original registration requests and related unicast (notification) responses MUST also be protected, including and especially the informative responses from the server. An exception is the case discussed in Appendix D, where the informative response from the server is not protected.

In order to protect unicast messages exchanged between the server and each client, including the original client registration (see Section 5), alternative security protocols than Group OSCORE can be used, such as OSCORE [RFC8613] and/or DTLS [RFC9147]. However, it is RECOMMENDED to use OSCORE or Group OSCORE, in order to reduce the number of libraries that the clients and the server have to support.

9.1. 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]. The mechanism defined in this section is OPTIONAL to support for the client and server.

In addition to what is defined in Section 4, the CBOR map in the informative response payload contains the following fields, whose CBOR abbreviations 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 with the Group Manager for the OSCORE group, encoded as a CBOR text string.

* Optionally, 'hkdf', with value the HKDF Algorithm used in the OSCORE group, encoded as a CBOR text string or integer. The HKDF Algorithm is specified by the HMAC Algorithm value, which is taken from the 'Value' column of the "COSE Algorithms" registry [COSE.Algorithms]. For example, the HKDF Algorithm HKDF SHA-256 is specified as the HMAC Algorithm HMAC 256/256.

* Optionally, 'cred_fmt', with value the format of the authentication credentials used in the OSCORE group, encoded as a CBOR integer. The value is taken from the 'Label' column of the "COSE Header Parameters" Registry [COSE.Header.Parameters]. Consistent with Section 2.4 of [I-D.ietf-core-oscore-groupcomm], acceptable values denote a format that provides the public key and a comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable).

  At the time of writing this specification, acceptable formats of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claim Sets (CCSs) [RFC8392], X.509 certificates [RFC5280], and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Further formats may be available in the future, and they would be acceptable to use as long as they comply with the criteria above.

  [As to C509 certificates, there is a pending registration requested by draft-ietf-cose-cbor-encoded-cert.]

* Optionally, 'gp_enc_alg', with value the Group Encryption Algorithm used in the OSCORE group to encrypt messages protected with the group mode, encoded as a CBOR text string or integer. The value is taken from the 'Value' column of the "COSE Algorithms" registry [COSE.Algorithms].

* Optionally, 'sign_alg', with value the Signature Algorithm used to sign 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, 'sign_params', encoded as a CBOR array and including the following two elements:

  - 'sign_alg_capab': a CBOR array, with the same format and value of the COSE capabilities array for the algorithm indicated in 'sign_alg', as specified for that algorithm in the 'Capabilities' column of the "COSE Algorithms" Registry [COSE.Algorithms].

- 'sign_key_type_capab': a CBOR array, with the same format and value of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'sign_alg', as specified for that key type in the 'Capabilities' column of the "COSE Key Types" Registry [COSE.Key.Types].

The values of 'sign_alg', 'sign_params', and 'cred_fmt' provide an early knowledge of the format of authentication credentials as well as of the type 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 authentication credential in the correct expected format and including a public key of the correct expected type, at the very first step of the (ACE) join process.

The values of 'hkdf', 'gp_enc_alg', and 'sign_alg' 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 corresponding fields are OPTIONAL in the informative response. However, the 'join_uri' and 'sec_gp' fields MUST be present if this mechanism is 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 5.1). If the client chooses not to, then 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.

9.2. Server-Side Requirements

When using Group OSCORE to protect multicast notifications, the server performs the operations described in Section 4, with the following differences.

9.2.1. Registration

The phantom registration request MUST be protected, by using Group OSCORE. In particular, the group mode of Group OSCORE defined in Section 7 of [I-D.ietf-core-oscore-groupcomm] MUST be used.

The server protects the phantom registration request as defined in Section 7.1 of [I-D.ietf-core-oscore-groupcomm] by using its Sender Context, i.e., like if it was the actual sender. 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). Consistent with that, the OSCORE Option value in the phantom registration request specifies:

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

* In the 'Partial IV' field, the Partial IV encoding the previously consumed Sender Sequence Number value SN of the server in the OSCORE group, i.e., (SN* - 1).

9.2.2. 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 9.2.1). Consequently, the following applies:

* The specified Code is 0.05 (FETCH) [RFC8132].

* The sequence of CoAP options will be limited to the outer, non encrypted options.

* A payload is always present, as the ciphertext followed by the countersignature.

Note that, in terms of transport-independent information, the registration request from the client typically differs from the phantom request. Thus, the server has to include the 'ph_req' parameter in the informative response. An exception is the case discussed in Appendix D.

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 9.2.3). This applies also to the initial multicast notification INIT_NOTIF built at Step 6 of Section 4.1.

Optionally, the informative response includes additional parameters that provide information about the OSCORE group to join (see Section 9.1).

9.2.3. 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 7 of [I-D.ietf-core-oscore-groupcomm] MUST be used.

The process described in Section 7.3 of [I-D.ietf-core-oscore-groupcomm] applies, with the following additions when building the two OSCORE external_aad structures to encrypt and sign the multicast notification (see Section 3.4 of [I-D.ietf-core-oscore-groupcomm]).

* The 'request_kid' element contains the value of the 'kid' field in the OSCORE Option value of the phantom registration request, i.e., the Sender ID of the server.

* The 'request_piv' element contains the value of the 'Partial IV' field in the OSCORE Option value of the phantom registration request, i.e., the Partial IV encoding the consumed Sender Sequence Number SN of the server.

* The 'request_kid_context' element contains the value of the 'kid context' field in the OSCORE Option value 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.

9.2.4. Cancellation

When canceling a group observation as defined in Section 4.5, the multicast response with error code 5.03 (Service Unavailable) is protected with Group OSCORE, as per Section 7.3 of [I-D.ietf-core-oscore-groupcomm]. The server MUST use its own Sender Sequence Number as Partial IV to protect the error response and MUST include the Partial IV in the OSCORE Option value of the response.

9.3. Client-Side Requirements

When using Group OSCORE to protect multicast notifications, the client performs as described in Section 5, with the following differences.

9.3.1. Informative Response

Upon receiving the informative response from the server, the client performs the same steps defined in Section 5.2, with the following additions.

When performing Step 2, the client expects the 'ph_req' parameter to be included in the informative response, which is otherwise considered malformed. An exception is the case discussed in Appendix D.

Once completed Step 2, the client decrypts and verifies the rebuilt phantom registration request as defined in Section 7.2 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 with 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', 'Partial IV', and 'kid context' fields from the OSCORE Option value 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 5.1). If the client chooses not to, then the client SHOULD explicitly withdraw from the group observation.

After successful decryption and verification, the client performs Step 3 in Section 5.2, considering the decrypted phantom registration request.

If the informative response includes the parameter 'last_notif', the client also decrypts and verifies the latest multicast notification rebuilt at Step 5 in Section 5.2, just like it would for the multicast notifications transmitted as CoAP messages on the wire (see Section 9.3.2). If decryption and verification succeed, the client proceeds with Step 6, considering the decrypted latest multicast notification. Otherwise, the client proceeds to Step 7.

9.3.2. Notifications

After having successfully processed the informative response as defined in Section 9.3.1, the client will decrypt and verify every multicast notification for the target resource as defined in Section 7.4 of [I-D.ietf-core-oscore-groupcomm], with the following difference.

For both decryption and signature verification, the client MUST set the external_aad structure defined in Section 3.4 of [I-D.ietf-core-oscore-groupcomm] as follows. The particular way to achieve this is implementation specific.

* The 'request_kid' element takes the value of the 'kid' field from the OSCORE Option value of the phantom registration request (see Section 9.3.1).

* The 'request_piv' element takes the value of the 'Partial IV' field from the OSCORE Option value of the phantom registration request (see Section 9.3.1).

* The 'request_kid_context' element takes the value of the 'kid context' field from the OSCORE Option value of the phantom registration request (see Section 9.3.1).

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

10. Example with Group OSCORE

The following example refers to two clients C1 and C2 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. The server 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:

* C1 and S have a pairwise OSCORE Security Context. In particular, C1 has 'kid' = 0x01 as Sender ID and SN_1 = 101 (i.e., 0x65) as Sender Sequence Number.

* C2 and S have a pairwise OSCORE Security Context. In particular, C2 has 'kid' = 0x02 as Sender ID and SN_2 = 201 (i.e., 0xc9) as Sender Sequence Number.

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

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

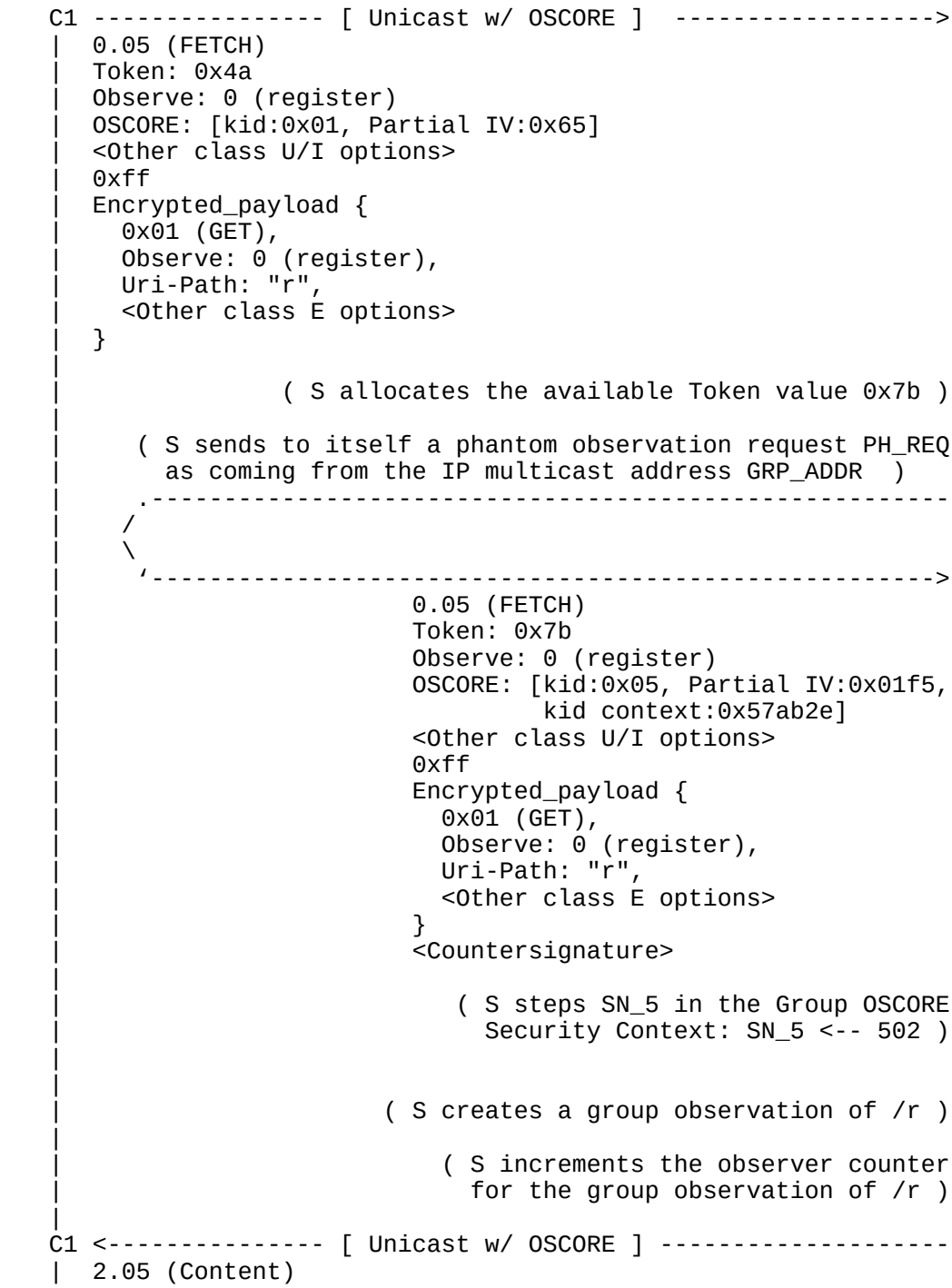
* The application-extension identifier "cri" defined in Section 3.4 of [I-D.ietf-cbor-edn-literals] is used to notate a CBOR Extended Diagnostic Notation (EDN) literal for a CRI.

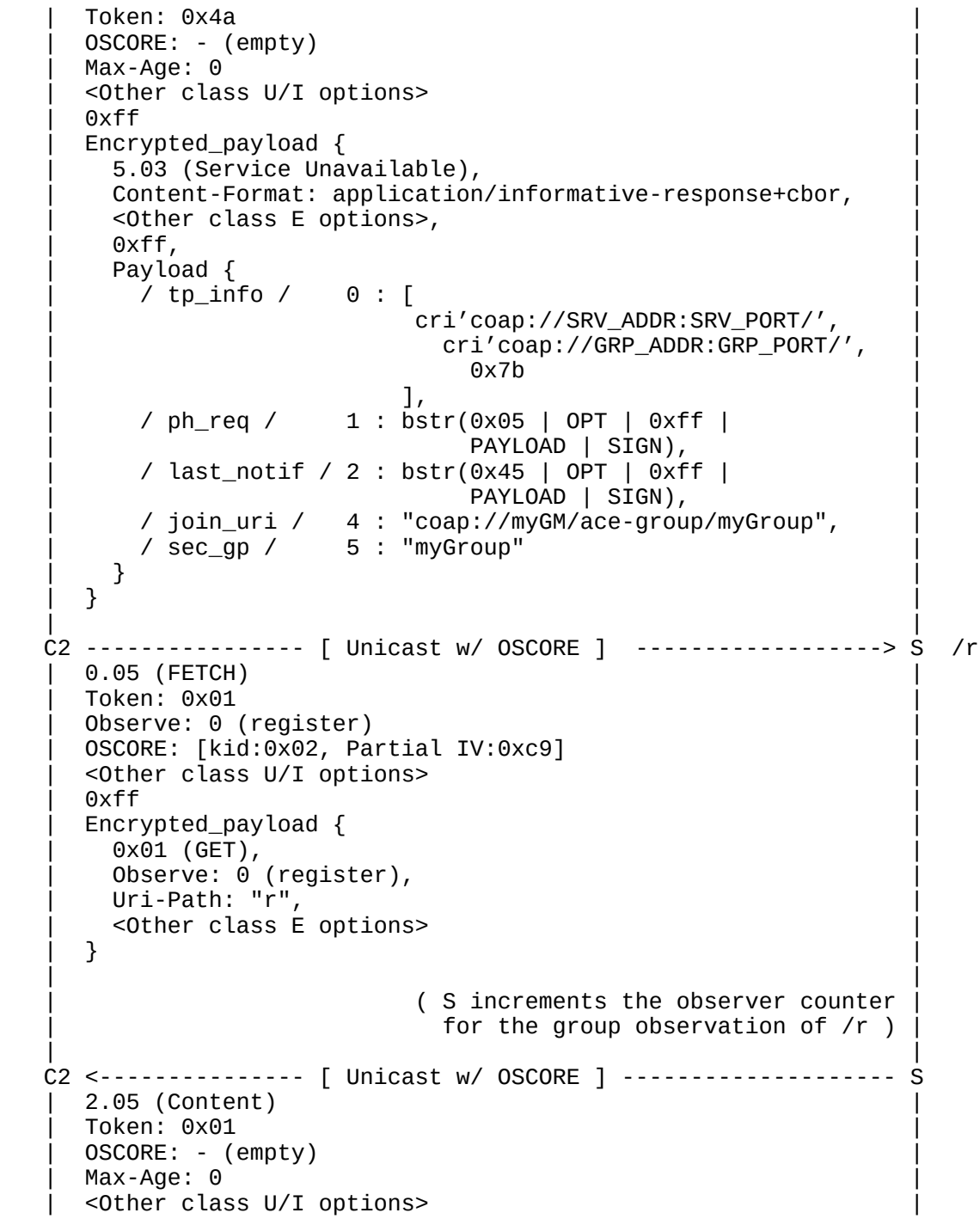
* 'bstr(X)' denotes a CBOR byte string with value the byte serialization of X, with '|' denoting byte concatenation.

* '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 countersignature 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.


C1 ---------- [Unicast w/ OSCORE] ----------> S /r
0.05 (FETCH)
Token: 0x4a
Observe: 0 (register)
OSCORE: [kid:0x01, Partial IV:0x65]
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
(S allocates the available Token value 0x7b)
(S sends to itself a phantom observation request PH_REQ
as coming from the IP multicast address GRP_ADDR)
/
\
'----------> /r
0.05 (FETCH)
Token: 0x7b
Observe: 0 (register)
OSCORE: [kid:0x05, Partial IV:0x01f5,
kid context:0x57ab2e]
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
<Countersignature>
(S steps SN_5 in the Group OSCORE
Security Context: SN_5 <-- 502)
(S creates a group observation of /r)
(S increments the observer counter
for the group observation of /r)
C1 <---------- [Unicast w/ OSCORE] ---------- S
2.05 (Content)


Token: 0x4a
OSCORE: - (empty)
Max-Age: 0
<Other class U/I options>
0xff
Encrypted_payload {
5.03 (Service Unavailable),
Content-Format: application/informative-response+cbor,
<Other class E options>,
0xff,
Payload {
/ tp_info / 0 : [
cri'coap://SRV_ADDR:SRV_PORT/',
cri'coap://GRP_ADDR:GRP_PORT/',
0x7b
],
/ ph_req / 1 : bstr(0x05 | OPT | 0xff |
PAYLOAD | SIGN),
/ last_notif / 2 : bstr(0x45 | OPT | 0xff |
PAYLOAD | SIGN),
/ join_uri / 4 : "coap://myGM/ace-group/myGroup",
/ sec_gp / 5 : "myGroup"
}
}
C2 [Unicast w/ OSCORE] S /r
0.05 (FETCH)
Token: 0x01
Observe: 0 (register)
OSCORE: [kid:0x02, Partial IV:0xc9]
<Other class U/I options>
0xff
Encrypted_payload {
0x01 (GET),
Observe: 0 (register),
Uri-Path: "r",
<Other class E options>
}
(S increments the observer counter
for the group observation of /r)
C2 [Unicast w/ OSCORE] S
2.05 (Content)
Token: 0x01
OSCORE: - (empty)
Max-Age: 0
<Other class U/I options>

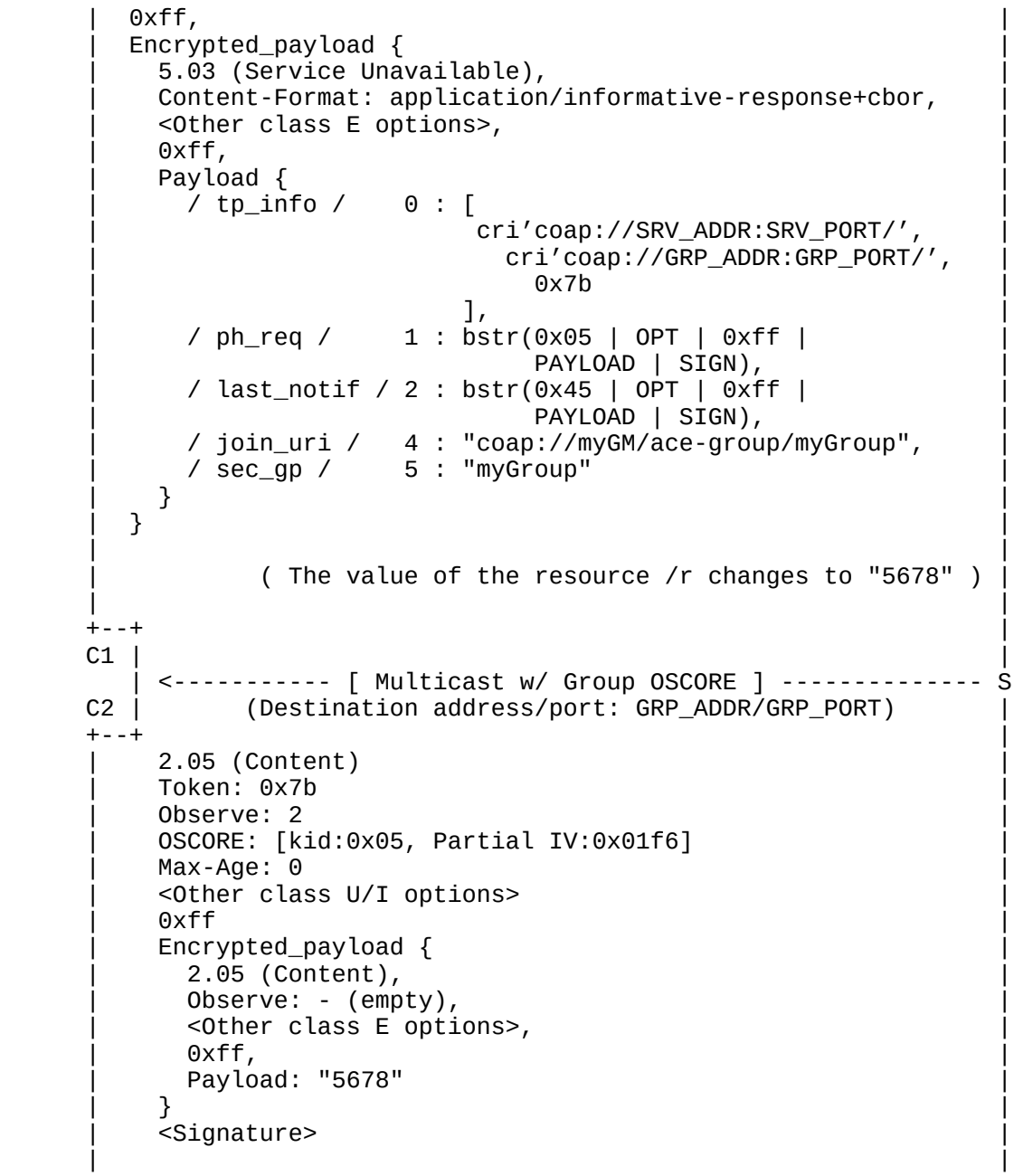


Figure 7: Example of Group Observation with Group OSCORE

The two external_aad structures used to encrypt and sign the multicast notification above include 'request_kid' = 0x05, 'request_piv' = 0x01f5, and 'request_kid_context' = 0x57ab2e. These values are specified in the 'kid', 'Partial IV', and 'kid context' fields of the OSCORE Option value 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 structures for decrypting and verifying this multicast notification and the following ones in the group observation.

11. Informative Response Parameters

This document defines a number of fields used in the informative response defined in Section 4.2.

The table below summarizes them and specifies the CBOR key to use as abbreviation, 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	0	array	Section 4.2 of [RFC-XXXX]
ph_req	1	byte string	Section 4.2 of [RFC-XXXX]
last_notif	2	byte string	Section 4.2 of [RFC-XXXX]
next_not_before	3	unsigned integer	Section 4.2 of [RFC-XXXX]
ending	4	integer or float	Section 4.2 of [RFC-XXXX]
join_uri	5	text string	Section 9.1 of [RFC-XXXX]
sec_gp	6	text string	Section 9.1 of [RFC-XXXX]
as_uri	7	text string	Section 9.1 of [RFC-XXXX]

hkdf	8	integer or text string	Section 9.1 of [RFC-XXXX]
cred_fmt	9	integer	Section 9.1 of [RFC-XXXX]
gp_enc_alg	10	integer or text string	Section 9.1 of [RFC-XXXX]
sign_alg	11	integer or text string	Section 9.1 of [RFC-XXXX]
sign_params	12	array	Section 9.1 of [RFC-XXXX]
gp_material	13	map	Appendix C of [RFC-XXXX]
srv_cred	14	byte string	Appendix C of [RFC-XXXX]
srv_identifier	15	byte string	Appendix C of [RFC-XXXX]
exi	16	unsigned integer	Appendix C of [RFC-XXXX]
exp	17	integer or float	Appendix C of [RFC-XXXX]

Table 2: Informative Response Parameters.

Note to RFC Editor: In the table above, please replace "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

12. Transport Protocol Information

Section 4.2.1.1 defines the transport-specific information that the server has to specify as elements of ’tpi_details’ within the ’tp_info’ parameter of the informative response defined in Section 4.2, when CoAP responses are transported over UDP.

Table 3 defines the corresponding entry that Section 14.6 registers in the "CoAP Transport Information" registry defined in this document.

CoAP Transport	Transport Information Details	Reference
CoAP over UDP	tpi_client,tpi_token	Section 4.2.1.1 of [RFC-XXXX]

Table 3: CoAP Transport Information for CoAP over UDP.

Note to RFC Editor: In the table above, please replace "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

13. Security Considerations

In addition to the security considerations from [RFC7252], [RFC7641], [I-D.ietf-core-groupcomm-bis], [RFC8613], and [I-D.ietf-core-oscore-groupcomm], the following considerations hold for this document.

13.1. Unprotected Communications

If communications are not protected, the server might not be able to effectively authenticate a new client when it registers as an observer. Section 7 of [RFC7641] specifies how, in such a case, the server must strictly limit the number of notifications sent between receiving acknowledgements from the client, as confirming to be still interested in the observation; i.e., any notifications sent in Non-confirmable messages must be interspersed with confirmable messages.

This is not possible to achieve by the same means when using the communication model defined in this document, since multicast notifications are sent as Non-confirmable messages. Nonetheless, the server might obtain such acknowledgements by other means.

For instance, the method defined in Section 8 to perform the rough counting of still interested clients triggers (some of) them to explicitly send a new observation request to acknowledge their interest. Then, the server can decide to terminate the group observation altogether, in the case that not enough clients are estimated to be still active.

If the method defined in Section 8 is used, the server SHOULD NOT send more than a strict number of multicast notifications for a given group observation, without having first performed a new rough counting of active clients. Note that, when using the method defined in Section 8 with unprotected communications, an adversary can craft and inject multiple new observation requests including the Feedback-Divider Option, hence inducing the server to overestimate the number of still interested clients and thus to inappropriately continue the group observation.

13.2. Protected Communications

If multicast notifications for an observed resource are protected using Group OSCORE as per Section 9, it is ensured that those are securely bound to the phantom registration request that started the group observation of that resource. Furthermore, the following applies.

* The original registration requests and related unicast (notification) responses MUST also be protected, including and especially the informative responses from the server. An exception is the case discussed in Appendix D, where the informative response from the server is not protected.

  Protecting informative responses from the server prevents on-path active adversaries from altering the conveyed IP multicast address and serialized phantom registration request.

* A re-registration request, possibly including the Feedback-Divider Option to support the rough counting of clients (see Section 8), MUST also be protected.

14. IANA Considerations

This document has the following actions for IANA.

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

14.1. Media Type Registrations

This document registers the media type "application/informative-response+cbor" for error messages as informative response defined in Section 4.2, 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 4.2 of [RFC-XXXX].

* Security considerations: See Section 13 of [RFC-XXXX].

* Interoperability considerations: N/A

* Published specification: [RFC-XXXX]

* 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 4.2 of [RFC-XXXX].

* Fragment identifier considerations: N/A

* Additional information: N/A

* Person & email address to contact for further information: CoRE WG mailing list (core@ietf.org) or IETF Applications and Real-Time Area (art@ietf.org)

* Intended usage: COMMON

* Restrictions on usage: None

* Author/Change controller: IETF

* Provisional registration: No

14.2. CoAP Content-Formats Registry

IANA is asked to add the following entry to the "CoAP Content-Formats" registry [CoAP.Content.Formats] within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

Content Type: application/informative-response+cbor

Content Coding: -

ID: TBD (value between 0 and 255)

Reference: [RFC-XXXX]

14.3. CoAP Option Numbers Registry

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

Number	Name	Reference
TBD18	Feedback-Divider	[RFC-XXXX]

Table 4: Registrations in the CoAP Option Numbers Registry

For the Feedback-Divider Option, the preferred value range is 0-255. In particular, 18 is the preferred option number.

Note to RFC Editor: In the table above, please replace TBD18 with the registered option number. Then, please delete this paragraph and the previous paragraph.

14.4. Target Attributes Registry

IANA is asked to register the following entry in the "Target Attributes" registry [Target.Attributes] within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

* Attribute Name: gp-obs
* Brief Description: Observable resource supporting group observation
* Change Controller: IETF
* Reference: Section 6 of [RFC-XXXX]

14.5. Informative Response Parameters Registry

This document establishes the "Informative Response Parameters" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

The registration policy is either "Private Use", "Standards Action with Expert Review", or "Specification Required" or "Expert Review" per [RFC8126]. "Expert Review" guidelines are provided in Section 14.7.

All assignments according to "Standards Action with Expert Review" are made on a "Standards Action" basis per Section 4.9 of [RFC8126] with "Expert Review" additionally required per Section 4.5 of [RFC8126]. The procedure for early IANA allocation of "standards track code points" defined in [RFC7120] also applies. When such a procedure is used, IANA will ask the designated expert(s) to approve the early allocation before registration. In addition, working group chairs are encouraged to consult the expert(s) early during the process outlined in Section 3.1 of [RFC7120].

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 of the item.

* CBOR Key: This is the value used as the CBOR map key of the item. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 as well as text strings of length 1 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535 as well as text strings of length 2 are designated as "Specification Required". Integer values greater than 65535 as well as text strings of length greater than 2 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* 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, if one exists.

This registry has been initially populated by the entries in Table 2.

14.6. CoAP Transport Information Registry

This document establishes the "CoAP Transport Information" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

The registration policy is "Expert Review" [RFC8126]. "Expert Review" guidelines are provided in Section 14.7.

The columns of this registry are:

* CoAP Transport: This field contains a text string. The value MUST be unique and it uniquely identifies the transport used for CoAP messages.

* Transport Information Details: This field contains a list of text strings, where two adjacent strings are separated by a single comma character. Each text string is the name of an element that provides transport-specific information related to a pertinent CoAP request. Optional elements are prepended by '?' and MUST be specified next to each other as last ones.

* Reference: This contains a pointer to the public specification for the item, if one exists.

This registry has been initially populated by the entry in Table 3.

14.7. Expert Review Instructions

"Standards Action with Expert Review", "Specification Required", and "Expert Review" are three of the registration policies defined for the IANA registries established in this document. 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 Action With Expert Review" range of point assignment. Specifications should exist for "Specification Required" ranges, but early assignment before a specification is available is considered to be permissible. 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

[CoAP.Content.Formats]
IANA, "CoAP Content-Formats", <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>.

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

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

[COSE.Header.Parameters]
IANA, "COSE Header Parameters", <https://www.iana.org/assignments/cose/cose.xhtml#header-parameters>.

[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. and F. Palombini, "Key Management for Group Object Security for Constrained RESTful Environments (Group OSCORE) Using Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-21, 14 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-21>.

[I-D.ietf-cbor-edn-literals]
Bormann, C., "CBOR Extended Diagnostic Notation (EDN)", Work in Progress, Internet-Draft, draft-ietf-cbor-edn-literals-22, 6 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-22>.

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

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

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

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

[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/rfc/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/rfc/rfc6838>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/rfc/rfc7120>.

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

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/rfc/rfc7595>.

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

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

[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/rfc/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/rfc/rfc8126>.

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

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

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

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

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

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

[RFC9203] Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "The Object Security for Constrained RESTful Environments (OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework", RFC 9203, DOI 10.17487/RFC9203, August 2022, <https://www.rfc-editor.org/rfc/rfc9203>.

[Target.Attributes] IANA, "Target Attributes", <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#target-attributes>.

15.2. Informative References

[I-D.ietf-core-cacheable-oscore] Amsüss, C. and M. Tiloca, "End-to-End Protected and Cacheable Responses for the Constrained Application Protocol (CoAP) using Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-cacheable-oscore-01, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-cacheable-oscore-01>.

[I-D.ietf-core-coap-pubsub] Jimenez, J., Koster, M., and A. Keränen, "A publish-subscribe architecture for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-19, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pubsub-19>.

[I-D.ietf-core-coral] Amsüss, C. and T. Fossati, "The Constrained RESTful Application Language (CoRAL)", Work in Progress, Internet-Draft, draft-ietf-core-coral-06, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coral-06>.

[I-D.ietf-core-multicast-notifications-proxy] Tiloca, M., Höglund, R., Amsüss, C., and F. Palombini, "Using Proxies for Observe Notifications as CoAP Multicast Responses", Work in Progress, Internet-Draft, draft-ietf-

core-multicast-notifications-proxy-00, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-multicast-notifications-proxy-00>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., H旦glund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-17, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-17>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Ams端ss, C., and P. Van der Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-19, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-discovery-19>.

[MOBICOM99]
Ni, S., Tseng, Y., Chen, Y., and J. 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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/rfc/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/rfc/rfc7519>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

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

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

[RFC9200] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth)", RFC 9200, DOI 10.17487/RFC9200, August 2022, <https://www.rfc-editor.org/rfc/rfc9200>.

[RFC9953] Lenders, M. S., Ams端ss, C., G端ndoト歛n, C., Schmidt, T. C., and M. Wテ、hlisch, "DNS over CoAP (DoC)", RFC 9953, DOI 10.17487/RFC9953, March 2026, <https://www.rfc-editor.org/rfc/rfc9953>.

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 (see Section 4.2), the server can also make the same group observation data available through different means, such as those described in Appendix A.1 and Appendix A.2.

In such a case, the server has to first start the group observation (see Section 4.1), before making the corresponding data available.

When distributed through different means than informative responses, the group observation data has to specify the time when the group observation is planned to be canceled by the server. In particular, the server commits to keeping the group observation ongoing until the scheduled cancellation time is reached. Before that time, the server might however retract the advertised group observation data and thus make it not available to new clients.

After a client has obtained the group observation data from different sources than an informative response, the client can simply set up the right multicast address and start receiving multicast notifications for the group observation. Consequently, the server will not receive an observation request from that client, will not follow-up with a corresponding informative response, and thus its observer counter (see Section 4) is not incremented to reflect the presence of the new client.

A.1. Topic Discovery in Publish-Subscribe Settings

In a Publish-Subscribe scenario (e.g., see [I-D.ietf-core-coap-pubsub]), a group observation can be discovered along with topic metadata.

To this end, together with the topic metadata, the server has to publish the same information associated with the group observation that would be conveyed in the informative response returned to observer clients (see Section 4.2).

This information especially includes the phantom observation request associated with the group observation, as well as the addressing information of the server and the addressing information where multicast notifications are sent to.

Figure 8 provides an example where a group observation is discovered. The example assumes a CoRAL namespace [I-D.ietf-core-coral] that contains properties analogous to those in the Content-Format "application/informative-response+cbor".

Note that the information about the transport protocol used for the group observation is not expressed through a dedicated element equivalent to 'tp_id' of the informative response (see Section 4.2.1). Instead, it is expressed through the scheme and authority components of the two URIs specified as 'tp_info_server' and 'tp_info_client', where the former specifies the addressing information of the server (like 'tpi_server' in Section 4.2.1.1), and the latter specifies the addressing information where multicast notifications are sent to (like 'tpi_client' in Section 4.2.1.1).

Request:

```
GET </ps/topics?rt=oic.r.temperature>
Accept: 65087 (application/coral+cbor)
```

Response:

```
2.05 Content
Content-Format: 65087 (application/coral+cbor)

rdf:type [ = <http://example.org/pubsub/topic-list>,
       topic [ = </ps/topics/1234>,
           tp_info_server <coap://[2001:db8::1]>,
           tp_info_client <coap://[ff35:30:2001:db8::123]>,
           tp_info_token "7b"^^xsd::hexBinary,
           ph_req "0160.."^^xsd::hexBinary,
           last_notif "256105.."^^xsd::hexBinary,
           ending "2051251201"^^xsd::unsignedLong,
       ]
]
```

Figure 8: 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 for the group observation in question.

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 switched off) cellular uplink to discover the topics it needs to update its display to, and then listen on a LoRaWAN 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. Figure 9 shows an example of a possible interface, as relying on the already known Token value of intercepted multicast notifications associated with 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 /    0 : [
                       [ / tpi_server /
                        -1, / scheme-id -- equivalent to "coap" /
                         h'20010db8000000000000000000000000ab' / host-ip /
                       ],
                       [ / tpi_client /
                        -1, / scheme-id -- equivalent to "coap" /
                        h'ff35003020010db800000000000000023', / host-ip /
                        61616 / port /
                       ],
                       h'7b' / tpi_token /
                      ],
   / ph_req /     1 : h'0160...528c', / elided for brevity /
   / last_notif / 2 : h'256105...4fa1', / elided for brevity /
   / ending /     4 : 2051251201

 }
```

Figure 9: Group Observation Discovery with Server Introspection

For example, a network sniffer could offer sending such a request when unknown multicast notifications are seen in the network. Consequently, it can associate those notifications with a URI, or decrypt them if it is a 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 8.

In particular, Appendix B.1 describes the algorithm for the client side and 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 FEEDBACK-DIVIDER 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(FEEDBACK-DIVIDER);
       opt.set(0);
       req.setOption(opt);

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

B.2. Client Side (Optimized Version)

```
   input:  int Q, // Value of the FEEDBACK-DIVIDER 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(FEEDBACK-DIVIDER);
       opt.set(0);
       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(FEEDBACK-DIVIDER);
   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
            received between t1 and t2, and including
            the FEEDBACK-DIVIDER Option with value 0>;

   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 set up 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 4.2).

In addition to what is defined in Section 4, the CBOR map in the informative response payload contains the following fields, whose CBOR abbreviations 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 6.3 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 [RFC9203].

  In particular, the following elements of the Group_OSCORE_Input_Material object are included, using the same CBOR abbreviations from the OSCORE Security Context Parameters Registry, as in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

  - 'ms', 'contexId', 'cred_fmt', 'gp_enc_alg', 'sign_alg', and 'sign_params'. These elements MUST be included.

  - 'hkdf' and 'salt'. These elements MAY be included.

  The 'group_senderId' element of the Group_OSCORE_Input_Material object MUST NOT be included.

Note that no information is provided as related to the AEAD Algorithm, or to the Pairwise Key Agreement Algorithm and its parameters. In fact, the clients and the server will never use the pairwise mode of Group OSCORE as per Section 8 of [I-D.ietf-core-oscore-groupcomm] and will not need to compute a cofactor Diffie-Hellman shared secret in this OSCORE group.

It follows that:

- In the Common Context of the Group OSCORE Security Context, the parameter AEAD Algorithm and the parameter Pairwise Key Agreement Algorithm are not set (see Section 2.1.1 of [I-D.ietf-core-oscore-groupcomm] and Section 2.1.10 of [I-D.ietf-core-oscore-groupcomm]).

- Aligned with the previous point, when building the two OSCORE external_aad structures to process messages protected with Group OSCORE in this OSCORE group, (see Section 3.4 of [I-D.ietf-core-oscore-groupcomm]), the elements 'alg_aead' and 'alg_pairwise_key_agreement' within the 'algorithms' arrays are set to the CBOR simple value null (0xf6).

* 'srv_cred': this element is a CBOR byte string, with value the original byte serialization of the server's authentication credential used in the OSCORE group. In particular, the original byte serialization complies with the format specified by the 'cred_fmt' element of 'gp_material'.

* 'srv_identifier': this element is encoded as a CBOR byte string, with value the Sender ID that the server has in the OSCORE group.

* 'exi': this element has as value the residual lifetime of the keying material of the OSCORE group specified in the 'gp_material' parameter, encoded as a CBOR unsigned integer.

  The value represents the residual lifetime of the keying material in seconds, i.e., the number of seconds between the current time at the server and the time when the keying material expires. Upon receiving the informative response containing the 'exi' parameter, a client determines the expiration time of the keying material by adding the number of seconds specified in the 'exi' parameter to its current time.

If the server has a reliable way to synchronize its internal clock with UTC, then the server includes also the following field:

* 'exp': this element has as value the expiration time of the keying material of the OSCORE group specified in the 'gp_material' parameter, encoded as a CBOR integer or as a CBOR floating-point number.

  The value is the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what is specified for NumericDate in Section 2 of [RFC7519].

If a client has a reliable way to synchronize its internal clock with UTC and the 'exp' parameter is present in the informative response, then the client MUST use the 'exp' parameter value as expiration time for the group keying material.

Note that the informative response does not require to include an explicit proof of possession of the server's private key. Although the server is also acting as Group Manager and a proof-of-possession evidence of the Group Manager's private key is included in a full-fledged Join Response (see Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore]), such proof of possession will be achieved through every multicast notification that the server sends, as protected with the group mode of Group OSCORE and including a signature computed with its private key.

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 the Security Context like a "silent server" would, i.e., without a Sender Context. From then on, the client uses the received keying material to process the incoming multicast notifications from the server.

Since the server is also acting as the Group Manager, the authentication credential of the server provided in the 'srv_cred' element of the informative response is also used in the 'gm_cred' element of the external_aad structure for encrypting and signing the phantom request and multicast notifications (see Section 3.4 of [I-D.ietf-core-oscore-groupcomm]).

Furthermore, 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 the phantom requests and the multicast notifications in group observations that it hosts, according to the method 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 that 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.

Upon expiration of the group keying material as indicated in the informative response by the 'exp' parameter (if present) and the 'exi' parameter:

* The server MUST stop using the keying material and MUST cancel the group observations for which that keying material is used (see Section 4.5 and Section 9.2.4). If the server creates a new group observation as a replacement or follow-up using the same OSCORE group, then the following applies:

  - The server MUST update the OSCORE Master Secret.

  - The server MUST update the ID Context used as Group Identifier (Gid), consistent with Section 12.2 of [I-D.ietf-core-oscore-groupcomm].

  - The server MAY update the OSCORE Master Salt.

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

Before the keying 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 keying material. In particular, while it is analogous to the informative response defined in Section 4.2, this response has the following differences:

* it additionally contains the parameters mentioned above, for the next group keying material to be used; and

* it MAY omit the 'tp_info' and 'ph_req' parameters, since the associated information is immutable throughout the observation lifetime.

The response has the same Token value T of the phantom registration request and it does not include an Observe Option. The server MUST use its own Sender Sequence Number as Partial IV to protect the error response and MUST include the Partial IV in the OSCORE Option value of the response.

When some clients leave the OSCORE group and forget about the group observation, the server does not have to provide the remaining clients with any stale Sender IDs, as normally required for Group OSCORE (see Section 12.2 of [I-D.ietf-core-oscore-groupcomm]). In fact, only two entities in the group have a Sender ID, i.e., the server and possibly the Deterministic Client, if the optimization defined in this appendix is combined with the use of phantom requests as Deterministic Requests (see Appendix D). In particular, both of them never change their Sender ID during the group lifetime and they both remain part of the group until the group ceases to exist.

As an alternative to renewing the keying material before it expires, the server can simply cancel the group observation (see Section 4.5 and Section 9.2.4), which results in the eventual re-registration of the clients that are still interested in the group observation.

Applications requiring backward security or 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 with information about the OSCORE group to join, as described in Section 9.1.

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, together with the transport-specific information required to listen to corresponding multicast notifications.

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). In either case, the server would already have started the group observation, before the associated phantom observation request was disseminated.

Then, the clients can set up the multicast address and group observation for listening to multicast notifications.

If Group OSCORE is used to protect the group observation (see Section 9) and the OSCORE group supports the concept of Deterministic Client [I-D.ietf-core-cacheable-oscore], then the server and clients in the OSCORE group can also independently compute the protected phantom observation request.

In such a case, the unprotected version of the phantom observation request can be made available to the clients as a smaller, plain CoAP message. As above, this can be pre-configured on the clients, or

they can obtain it through dedicated means (see Appendix A). In either case, the clients and the server can independently protect the plain CoAP message by using the approach defined in Section 3 of [I-D.ietf-core-cacheable-oscore], thus all computing the same protected Deterministic Request. The latter is used as the actual phantom observation request that the protected multicast notifications will match under the group observation in question.

When receiving the Deterministic Request, the server can clearly understand what is happening. In fact, as the result of an early check, the server recognizes the phantom request among the stored ones. This relies on a byte-by-byte comparison of the incoming message minus the transport-related fields, i.e., by considering only: i) the outer REST code; ii) the outer options; and iii) the ciphertext from the message payload.

If the server recognizes the received Deterministic Request as one of its self-generated deterministic phantom requests, then the server does not perform any Group OSCORE processing on it. This opens for replying with an unprotected response, although not indicating any OSCORE-related error. In particular, the server MUST reply with an informative response that MUST NOT be protected.

Note that the phantom registration request is, in terms of transport-independent information, identical to the same Deterministic Request sent by any client that wishes to take part in the group observation. Thus, if the server receives such a phantom registration request, the informative response may omit the 'ph_req' parameter (see Section 4.2). If a client receives an informative response that includes the 'ph_req' parameter and this specifies transport-independent information different from the one of the sent Deterministic Request, then the client considers the informative response malformed.

When using a Deterministic Request as a phantom observation request, the observer counter at the server (see Section 4) is not reliably incremented when new clients start participating in the group observation.

That is, the clients can simply set up the right multicast address and port number, and then start listening to multicast notifications bound to the Deterministic Request. Hence, the observer counter at the server is not incremented as new clients start listening to multicast notifications.

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

The setting described above requires the server and all the clients interested in taking part in the group observation to support the approach defined in [I-D.ietf-core-cacheable-oscore]. On the other hand, its use allows clients to start from a smaller, unprotected version of the phantom observation request, which does not need to be verified as a request generated by the server. Therefore, in applications where the OSCORE group is configured for the use of Deterministic Requests and observer clients are expected to support the approach defined in [I-D.ietf-core-cacheable-oscore], servers that start group observations are encouraged to do so by using Deterministic Requests as corresponding phantom observation requests.

If the optimization defined in Appendix C is also used, the 'gp_material' element in the informative response from the server MUST also include the following elements from the Group_OSCORE_Input_Material object.

* 'alg', as per Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'det_senderId' and 'det_hash_alg', defined in Section 4 of [I-D.ietf-core-cacheable-oscore]. These specify the Sender ID of the Deterministic Client in the OSCORE group and the hash algorithm used to compute Deterministic Requests (see Section 3.4.1 of [I-D.ietf-core-cacheable-oscore]).

Note that, like in Appendix C, no information is provided as related to the Pairwise Key Agreement Algorithm and its parameters. In fact, the clients and the server will not need to compute a cofactor Diffie-Hellman shared secret in this OSCORE group. It follows that:

* In the Common Context of the Group OSCORE Security Context, the parameter Pairwise Key Agreement Algorithm is not set (see Section 2.1.10 of [I-D.ietf-core-oscore-groupcomm]).

* Aligned with the previous point, when building the two OSCORE external_aad structures to process messages protected with Group OSCORE in this OSCORE group, (see Section 3.4 of [I-D.ietf-core-oscore-groupcomm]), the element 'alg_pairwise_key_agreement' within the 'algorithms' arrays is set to the CBOR simple value null (0xf6).

If a Deterministic Request is used as a phantom observation request for a group observation, the server does not assist clients that are interested in taking part in the group observation but do not support Deterministic Requests. This is consistent with the fact that the setup in question already relies on a lot of agreed pre-configuration.

Therefore, the following holds when a group observation for a target resource relies on a Deterministic Request as a phantom observation request.

* Every client that is interested to take part in such a group observation: has to support Deterministic Requests; and has to know the phantom observation request, as a result of pre-configuration or following its retrieval through dedicated means (see Appendix A).

* The server does not simultaneously run a parallel group observation for the same target resource, as associated with a different phantom observation request and intended to clients that do not support Deterministic Requests.

* If the server receives an observation request for the target resource that differs from the specific Deterministic Request associated with the group observation for that target resource, then the server replies as usual with an informative response, including: the transport-specific information, the phantom request (i.e., the expected Deterministic Request), and (optionally) the latest notification.

Appendix E. Document Updates

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

E.1. Version -13 to -14

* Clarified expected roles in the OSCORE group for clients and server.

* The HKDF Algorithm of Group OSCORE is specified as the corresponding HMAC Algorithm.

* Generalized semantics of the Multicast-Response-Feedback-Divider Option and renamed it as Feedback-Divider.

* Clarified pre-conditions and advantages of using deterministic phantom requests.

* Removed unnecessary normative language.

* Fixes and simplifications in the example with Group OSCORE.

* Updated references.

* Clarifications and editorial improvements.

E.2. Version -12 to -13

* Content on proxies moved out to the new document draft-ietf-core-multicast-notifications-proxy.

* Clarified avoidance of link-local addresses.

* Alignment with the update to the "Uniform Resource Identifier (URI) Schemes" IANA registry made by draft-ietf-core-href-26.

* Revised value syntax for the 'Transport Information Details' column of the new IANA registry "CoAP Transport Information".

* Suggested range and value for the registration in the IANA registry "CoAP Option Numbers".

* Updated references.

* Editorial improvements.

E.3. Version -11 to -12

* Changed CBOR type of 'ending' and 'exp' to be integer or float.

* Avoid limiting the informative response to this protocol.

* Transport identified by (scheme-id, authority) in the CRI of 'tpi_server'.

* Renamed 'sign_enc_alg' to 'gp_enc_alg', now aligned with draft-ietf-ace-key-groupcomm-oscore.

* Editorial improvements.

E.4. Version -10 to -11

* Do not rule out original observation requests sent over multicast.

* Defined 'ending' parameter for the informative response payload.

* Group observation data available on different sources can be removed.

* Initial description of a scenario with a reverse-proxy.

* Minor fixes in examples.

* Clarifications and editorial improvements.

E.5. Version -09 to -10

* Fixed section numbers of referred documents.

* Revised registration policies in the IANA considerations.

* Clarifications and editorial improvements.

E.6. Version -08 to -09

* Revised ’tp_info’ also to use CRIs for transport information.

* Section restructuring: impact from proxies on rough counting of clients.

* Revised and repositioned text on deterministic phantom requests.

* Fixes in the examples with message exchanges.

* Clarifications and editorial improvements.

E.7. Version -07 to -08

* Fixed the CDDL definition ’srv_addr’ in ’tp_info’.

* Early mentioning that ’srv_addr’ cannot instruct redirection.

* The replay protection of multicast notifications is as per Group OSCORE.

* Consistently use the format uint for the Multicast-Response-Feedback-Divider Option.

* Fixed consumption of proxy-related options in a ticket request sent to the proxy.

* Possible use of the option Proxy-Cri or Proxy-Scheme-Number in a ticket request.

* Explained non-provisioning of some parameters in self-managed OSCORE groups.

* Use of 'exi' for relative expiration time in self-managed OSCORE groups.

* Improved notation in the examples of message exchanges with proxy.

* Clarifications and editorial improvements.

E.8. Version -06 to -07

* Added more details on proxies that do not support the Multicast-Response-Feedback-Divider Option.

* Added more details on the reliability of the client rough counting.

* Added more details on the unreliability of counting new clients, when the phantom request is obtained from other sources or is an OSCORE Deterministic Request.

* Revised parameter naming.

* Fixes in IANA considerations.

* Editorial improvements.

E.9. Version -05 to -06

* Clarified rough counting of clients when a proxy is used

* IANA considerations: registration of target attribute "gp-obs"

* Editorial improvements.

E.10. Version -04 to -05

* If the phantom request is an OSCORE Deterministic Request, there is no parallel group observation for clients that lack support.

* Clarification on pre-configured clients.

* Clarified early publication of phantom request.

* Fixes in IANA considerations.

* Fixed example with proxy and Group OSCORE.

* Editorial improvements.

E.11. Version -03 to -04

* Added a new section on prerequisites.

* Added a new section overviewing alternative variants.

* Consistent renaming of ’cli_addr’ to ’cli_host’ in ’tp_info’.

* Added pre-requirements for early retrieval of phantom request.

* Revised example with early retrieval of phantom request.

* Clarified use, rationale and example of phantom request as Deterministic Request.

* Editorial improvements.

E.12. Version -02 to -03

* Distinction between authentication credential and public key.

* Fixed processing of informative response at the client, when Group OSCORE is used.

* Discussed scenarios with pre-configured address/port and Token value.

E.13. Version -01 to -02

* Clarifications on client rough counting and IP multicast scope.

* The ’ph_req’ parameter is optional in the informative response.

* New parameter ’next_not_before’ for the informative response.

* Optimization when rekeying the self-managed OSCORE group.

* Security considerations on unsecured multicast notifications.

* Protection of the ticket request sent to a proxy.

* RFC8126 terminology in IANA considerations.

* Editorial improvements.

E.14. Version -00 to -01

* Simplified cancellation of the group observation, without using a phantom cancellation request.

* Aligned parameter semantics with core-oscore-groupcomm and ace-key-groupcomm-oscore.

* Clarifications about self-managed OSCORE group and use of Deterministic Requests for cacheable OSCORE.

* New example with a proxy, Group OSCORE and a deterministic phantom request.

* Fixes in the examples and editorial improvements.

Acknowledgments

The authors sincerely thank Carsten Bormann, Klaus Hartke, Jaime Jimテゥnez, Matthias Kovatsch嫩ゥ, John Preuテ Mattsson, Jim Schaad, Ludwig Seitz, and G旦ran Selander for their comments and feedback.

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

Authors’ Addresses

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

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

Christian Ams端ss
Hollandstr. 12/4
1020 Vienna
Austria
Email: christian@amsuess.com

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