Observe Notifications as CoAP Multicast Responses

draft-ietf-core-observe-multicast-notifications-09

Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF 120 meeting – Vancouver – July 24th, 2024
Recap

› **Observe notifications as multicast responses**
  – Many clients observe the same resource on a server (e.g., pub-sub)
  – Improved performance due to multicast delivery
  – Clients configured by the server, with a 5.03 error informative response

› **Token space managed by the server**
  – The Token space **belongs** to the group (clients)
  – The group **entrusts** the management to the server
  – All clients in a group observation use the same Token value

› **Multicast notifications bound to a Phantom Observation Request**
  – By means of the same Token value for that observation

› **Group OSCORE to protect multicast notifications**
  – The server aligns all clients of an observation on a same *external_aad*
  – All notifications for a resource are protected with that *external_aad*
Latest updates

Various editorial fixes, improvements, and reference updates

Clarifications, considerations, and fixes (1/2)

› Clients can be pre-configured for listening to multicast notifications
  › Still useful to send the regular observation request, possibly with No-Response:16
  › This helps the server keep count of the active observer clients

› Rough counting of active observer clients: discussed accuracy and reliability
  › More details on the impact due to proxies (with section restructuring)
  › More details on the impact due to the Phantom Request being a Deterministic Request [1]

› Consistent use of the format *uint* for the Multicast-Response-Feedback-Divider Option
  › The value 0 is encoded as the zero-length value

› Early mentioning about the 5.03 error informative response and its content
  › The source addressing information for the Phantom Request cannot instruct redirection

Latest updates

Clarifications, considerations, and fixes (2/2)

› Secured multicast notifications: the replay protection is as per Group OSCORE
  › No need for restating; removed pointer to RFC 8613

› Fixed text about the proxy “consuming” proxy-related options (e.g., Proxy-Scheme, …)
  › Relevant when Group OSCORE is used, and clients send a ticket request to the proxy

› Appendix C – Server self-managing the OSCORE group
  › More details on why some Group OSCORE parameters are not needed to be provided

› Appendix D – Use of Deterministic Requests [1] as Phantom Requests
  › Revised, brought up text on how the server handles Deterministic Phantom Requests

› Revised parameter naming, aligned with the naming in Group OSCORE

Latest updates

Protocol behavior (1/2)

› Appendix A – Early, public distribution of the Phantom Observation Request
  › The server can rely on means other than the 5.03 error informative response. If so, …
  › The server first starts the group observation, then makes the corresponding data available

› Appendix C – Server self-managing the OSCORE group
  › Use of the parameter 'exi' for relative expiration time of the OSCORE group

› Appendix D – If the Phantom Observation Request is a Deterministic Request [1] …
  › The server does not assist clients that do not support Deterministic Requests
  › No "twin" group observation based on a non-deterministic Phantom Observation Request

› Multicast-Response-Feedback-Divider Option, used for the rough counting of clients
  › More details on how a proxy reacts if receiving the option and not supporting it

› Mentioned possible use of the new options Proxy-Cri and Proxy-Scheme-Number [2]
Latest updates

› Protocol behavior (2/2) – Major change, discussed at IETF 114
  › Revised the ‘tp_info’ information bundle in the 5.03 error informative response
  › This meant **switching to using CRIs** [2] to encode transport-specific information

```php
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'
}
```

› First proposed in the PR #13
  › [https://github.com/core-wg/observe-multicast-notifications/pull/13](https://github.com/core-wg/observe-multicast-notifications/pull/13)

› Fully specified in the PR #14 [4], now merged
  › [https://github.com/core-wg/observe-multicast-notifications/pull/14](https://github.com/core-wg/observe-multicast-notifications/pull/14)

---

Use of CRIs in ‘tp_info’ – Details in Sections 4.2.1 and 4.2.1.1 of version -09

OLD approach

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

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

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

NEW approach in version -09

```plaintext
tp_info = [  
  tpi_server, ; Addressing information of the server  
  ? tpi_details ; Additional information about the request  
]

tpi_server = CRI ; From draft-ietf-core-href, with no local part

tpi_details = (  
  + elements ; Number, format, and encoding based on the ; scheme-id of the CRI in ‘tpi_server’  
)```
Use of CRIs in ‘tp_info’ – Details in Sections 4.2.1 and 4.2.1.1 of version -09

Old approach

\[
\text{tp_info} = [ \\
\text{tp_id : 1,} \quad \text{; UDP as transport protocol} \\
\text{srv_host : #6.260(bstr),} \quad \text{; Src. address of multicast notifications} \\
\text{srv_port : uint,} \quad \text{; Src. port of multicast notifications} \\
\text{token : bstr,} \quad \text{; Token value of the Phantom Request and} \\
\text{cli_host : #6.260(bstr),} \quad \text{; Dst. address of multicast notifications} \\
? \text{cli_port : uint} \quad \text{; Dst. port of multicast notifications} \\
] 
\]

New approach in version -09

\[
\text{tp_info_coap_udp} = [ \\
\text{tpi_server} \quad \text{; Addressing information of the server,} \\
\text{; as a CRI with scheme-id = -1 (coap)} \\
\text{; and with no local part} \\
\text{tpi_details_udp} \quad \text{; Additional information about the request,} \\
\text{; when CoAP over UDP is used} \\
] 
\]

\[
\text{tpi_details_udp} = ( \\
\text{tpi_client : CRI,} \quad \text{; Addressing information of the clients,} \\
\text{; as a CRI with scheme-id = -1 (coap)} \\
\text{; and with no local part --- Used as} \\
\text{; destination of multicast notifications} \\
\text{tpi_token : bstr} \quad \text{; Token value of the Phantom Request and} \\
\text{; of the associated multicast notifications} \\
) 
\]
Latest updates

› Security considerations
  › Rough counting of clients when communications are unprotected or protected

› Examples of message exchanges
  › Fixes; more details; improved notation; use of AASVG
  › Aligned with the new use of CRIs in the 5.03 error informative response

› IANA considerations
  › Fixed details of some registrations (e.g., Media Type)
  › Registration of the target attribute “gp-obs” (like “obs”, but for group observations)
  › Revised definition and pre-population of the new registry "Transport Protocol Indication"
    › || Scheme ID || URI Scheme Name || Transport Information Details || Reference ||
Next steps

› Describe how this works with a reverse-proxy
  – Related to the Github issue #4

› Consider the case where original Observe requests are sent over multicast

› Define the server behavior on terminating a group observation …
  – … whose Phantom Observation Request was publicly advertised. Request revocation?

› Define how SCHC compression should work for the two new CoAP options
  – Listen-To-Multicast-Responses
  – Multicast-Response-Feedback-Divider

› Need for reviews – Previously promised: Göran, Esko, Jaime, Carsten, Thomas
Thank you!

Comments/questions?

https://github.com/core-wg/observe-multicast-notifications
Backup
 Phantom request and error response

› The server requests the observation on its own, e.g., when:
  1. A first traditional registration request comes from a first client; or
  2. Some threshold is crossed – clients can be shifted to a group observation

› Consensus on Token & external_aad, by using a phantom observation request
  – Generated inside the server, it does not hit the wire
  – Like if sent by the group, from the multicast IP address of the group
  – Multicast notifications are responses to this phantom request

› The server sends to clients a 5.03 error informative response with:
  – Transport-specific information, e.g., the IP multicast address where notifications are sent to
  – The serialization of the phantom observation request (optional)
  – The serialization of the latest multicast notification (optional)
  – Minimum amount of time after which the next multicast notification will be sent (optional)
Server side

1. Build a GET phantom request; Observe option set to 0

2. Choose a value \( T \), from the Token space for messages …
   \(-\) … coming from the multicast IP address and addressed to the target resource

3. Process the phantom request
   \(-\) As coming from the group and its IP multicast address
   \(-\) As addressed to the target resource

4. Hereafter, use \( T \) as token value for the group observation

5. Store the phantom request, store (not send) the reply as \('last_notif'\)
Interaction with clients

- The server sends to new/shifted clients an **error informative response** with
  - ‘tp_info’: transport-specific information
    - ‘tpi_server’: source addressing information of the multicast notifications (as a CRI)
    - ‘tpi_client’: destination addressing information of the multicast notifications (as a CRI)
    - ‘tpi_token’: the selected Token value T, used for ‘ph_req’ and the multicast notifications
  - ‘ph_req’: serialization of the phantom request
  - ‘last_notif’: serialization of the latest sent multicast notification for the target resource
  - ‘next_not_before’: minimum amount of time after which the next multicast notification will be sent

- When the value of the target resource changes:
  - The server sends an Observe notification to the multicast IP address corresponding to ‘tpi_client’
  - The multicast notification has the Token value T of the phantom request

- When getting the error informative response, a client:
  - Configures an observation for an endpoint associated with the multicast IP address
  - Accepts observe notifications with Token value T, sent to that multicast IP address
C1 registration

C1

[ Unicast ]

S /r

GET
Token: 0x4a
Observe: 0 (register)
Uri-Path: "r"
<Other options>

(S allocates the available Token value 0x7b)

(S sends to itself a phantom observation request PH_REQ as coming from the IP multicast address GRP_ADDR)

GET
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 registration

```
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
    ],
    /lastnotif/2: bstr(0x45 | OPT | 0xff | PAYLOAD)
  }
```
C2 registration

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
    ],
    /lastnotif / 2 : bstr(0x45 | OPT | 0xff | PAYLOAD)
}
Multicast notification

( The value of the resource /r changes to "5678" )

C1

[ Multicast ]

C2

( Destination address/port: GRP_ADDR, GRP_PORT )

S

2.05

Token: 0x7b
Observe: 11
<Other options>
Payload: "5678"

› Same Token value of the Phantom Request

› Enforce binding between
  – Every multicast notification for the target resource
  – The (group) observation that each client takes part in
Security with Group OSCORE

› The phantom request is protected with Group OSCORE
  – $x$: the Sender ID (‘kid’) of the Server in the OSCORE group
  – $y$: the current SN value (‘piv’) used by the Server in the OSCORE group
  – $z$: the Group ID (‘kid_context’) used in the OSCORE group
  – Note: the Server consumes the value $y$ and does not reuse it as SN in the group

› To secure/verify all multicast notifications, the OSCORE $external_aad$ is built with:
  – ‘request_kid’ = $x$
  – ‘request_piv’ = $y$
  – ‘request_kid_context’ = $z$

› The phantom request is still included in the informative response
  – Each client retrieves $x$, $y$, and $z$ from the OSCORE Option value
Security with Group OSCORE

In the error response, the server can optionally specify also:

- ‘join_uri’ : Link to the Group Manager to join the OSCORE group
- ‘sec_gp’ : Name of the OSCORE group
- ‘as_uri’ : Link to the ACE Authorization Server associated to the Group Manager
- ‘hkdf’ : HKDF Algorithm
- ‘cred_fmt’ : Format used in the OSCORE group for the authentication credentials
- ‘gp_enc_alg’ : Group Encryption Algorithm (for encryption with the group mode)
- ‘sign_alg’ : Signature Algorithm
- ‘sign_params’ : Parameters of the Signature Algorithm and signing key
  - ‘sign_alg_capab’ : COSE capabilities of the ‘sign_alg’ algorithm
  - ‘sign_key_type_capab’ : COSE capabilities of the keys used by ‘sign_alg’
C1 registration w/ security

0.05 (FETCH)
Token: 0x4a
OSCORE: {kid: 0x01; piv: 101; ...}
<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 as coming from the IP multicast address )

0.05 (FETCH)
Token: 0x7b
OSCORE: {kid: 0x05 ; piv: 501;
  kid context: 0x57ab2e; ...}
<Other class U/I options>
0xff
Encrypted_payload {
  0x01 (GET),
  Observe: 0 (register),
  Uri-Path: "r",
  <Other class E options>
}
<Signature>

( S steps SN_5 in the Group OSCORE Security Context: SN_5 ← 502 )

( S creates a group observation of /r )
( S increments the observer counter for the group observation of /r )

The server protects the Phantom Request with Group OSCORE, using its Sender Context, as if it was the sender.
C1 registration w/ security

C1 registration w/ security

2.05 (Content)
Token: 0x4a
OSCORE: { piv: 301; ...
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\n    ],
    / 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" 
  }
}

0x05: Sender ID (‘kid’) of S in the OSCORE group
501: Sequence Number of S in the OSCORE group when S created the group observation
C2 registration w/ security

0x05: Sender ID ('kid') of S in the OSCORE group
501: Sequence Number of S in the OSCORE group when S created the group observation
Multicast notification w/ security

When encrypting and signing the multicast notification:
- The `external_aad` has `request_kid` = \texttt{0x05}, `request_iv` = \texttt{501} and `request_kid_context` = \texttt{0x57ab2e}
- Same for all following notifications for the same resource

Enforce secure binding between
- Every multicast notification for the target resource
- The (group) observation that each client takes part in
Support for intermediary proxies

› How it works
  – The proxy (next to the server) directly listens to the IP multicast address
  – The original Token of the phantom request has to match at the proxy
  – The proxy forwards multicast notifications back to each client
    › The proxy uses the Token values offered by the clients

› Without end-to-end security (Section 11)
  – The proxy can retrieve the phantom request from the informative response
  – No need to forward the informative response back to the clients

› With end-to-end security (Section 12)
  – The informative response is also protected with OSCORE or Group OSCORE
  – The proxy **cannot** retrieve the phantom request from the informative response
  – Each client has to explicitly provide the phantom request to the proxy
  – Exception: the phantom request is a Deterministic Request (see **-core-cachable-oscore**

---

draft-ietf-core-observe-multicast-notifications | IETF 120 Meeting | 2024-07-24 | Page 26