# Constrained RESTful Environments WG (core)

Chairs:

Jaime Jiménez <jaime.jimenez@ericsson.com>

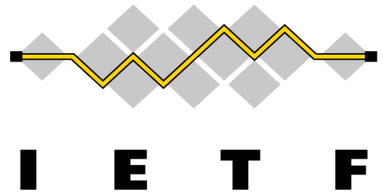
Marco Tiloca <marco.tiloca@ri.se> New!

Mailing list:

core@ietf.org

Jabber:

core@jabber.ietf.org



- We assume people have read the drafts
- Meetings serve to advance difficult issues by making good use of face-to-face communications
- Note Well: Be aware of the IPR principles, according to RFC 8179 and its updates

```
[ ] Blue Sheets
[ ] Jabber Scribe(s)
[ ] Note Taker(s)
```

# Note Well

**Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:**

- **The IETF plenary session**
- **The IESG, or any member thereof on behalf of the IESG**
- **Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices**
- **Any IETF working group or portion thereof**
- **Any Birds of a Feather (BOF) session**
- **The IAB or any member thereof on behalf of the IAB**
- **The RFC Editor or the Internet-Drafts function**

**All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.**

**Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.**

**A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.**

**A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.**

https://www.ietf.org/about/note-well/

All times are in UTC

# Wednesday (120 min)

- 13:00–13:10 Intro, Agenda, Status
- 13:10–13:25 CoRECONF
- 13:25–14:25 GroupComm
- 14:25–15:00 SenML

All times are in UTC

# Thursday (90 min)

- 13:00–13:05 Intro, Agenda
- 13:05–14:20 CoRE Applications
- 14:20–14:30 Flextime

# Agenda Bashing

# Intro

# Practicalities

- CoRE Interim meetings to occur every other week from the 29th of April. Time will be 14:00 UTC.
- We cleaned up the Github landing page at: core-wg.github.io
- Use of queuing at core@jabber.ietf.org
  - **`q+`** to add yourself to queue.
  - Otherwise use **`q+`** on Webex.
  - Use **`help q`** to request the list of commands.

multipart-ct-04 → RFC 8710 !! ✓

published 2020-02

hop-limit-07 → RFC 8768 !! ✓

published 2020-03

# RFC-Editor Queue

- draft-ietf-core-senml-etch-07
- draft-ietf-core-senml-more-units-06

# IESG Processing

- draft-ietf-core-resource-directory-24
  In Last Call
- draft-ietf-core-stateless-05
  In Last Call

# In Post-WGLC processing

- draft-ietf-core-echo-request-tag-09
  - WGLC to be formally closed

# WGLC to be issued

- draft-ietf-core-dev-urn-04
  - WGLC to be formally started

# CoRECONF


# CORECONF

Andy Bierman
Michel Veillette
Peter van der Stok
Alexander Pelov
Ivaylo Petrov

# Status sid-12

WGLC resulted in a good amount of editorial changes and some extra issues:

- Laurent Toutain and Andy Bierman believe it is ready
- Comments from *Peter van der Stork*, *Esko Dijk*, *Juergen Schoenwaelder*, *Michael Richardson*, *Tom Petch*
  - Prepare SID system for eventual change of YANG semantics
  - Concerns about Early Allocation
  - IANA Considerations group name
  - Other editorial or minor issues
- Some remarks are still not processed

# Status yang-cbor-12

WGLC resulted in a good amount of editorial changes and some extra issues:

- Laurent Toutain and Andy Bierman believe it is ready
- Comments from *Esko Dijk, Juergen Schoenwaelder*
  - Is there ever going to be another SID specification [JS]
  - Other editorial or minor issues
- All remarks are incorporated in master

# Status comi-09

WGLC resulted in a good amount of editorial changes and some extra issues:

- Laurent Toutain and Andy Bierman believe it is ready
- Comments from *Michael Richardson*
  - Naming of the draft cluster vs the protocol itself (also from other reviewers)
  - Security considerations

# Status yang-library-01


WGLC resulted in a good amount of editorial changes and some extra issues:

- Andy Bierman believe it is ready
- Comments *Tom Petch, Michael Richardson*
  - Security considerations
  - Other editorial changes and questions

# Timeline

To be discussed!

Likely:

- More discussion as needed and authors process comments
- Second WGLC
- Ship to IESG around end of April

# GroupComm

# Group Communication for the Constrained Application Protocol (CoAP)

draft-ietf-core-groupcomm-bis-00

**Esko Dijk, IoTconsultancy.nl**
Chonggang Wang, InterDigital
Marco Tiloca, RISE

IETF CoRE WG virtual interim, April 8th, 2020

# Goal

- Intended normative successor of experimental RFC 7390 (if approved)
  - As a Standards Track document
  - Obsoletes RFC 7390, Updates RFC 7252 / 7641
- Be standard reference for implementations that are now based on RFC 7390, e.g.:
  - “Eclipse Californium 2.0.x” (Eclipse Foundation)
  - “Implementation of CoAP Server & Client in Go” (OCF)
- What’s in scope?
  - CoAP group communication over UDP/IP, including latest developments (Observe/Blockwise/Security …)
  - Unsecured CoAP or group-OSCORE-secured communication
  - Principles for secure group configuration
  - Use cases (appendix)

# Groupcomm-bis-03/00: process view

- Updated with reviewers' comments (Jim [1], Thomas [2])
- Adopted as CoRE WG document
  - draft-dijk-core-groupcomm-bis-03 (March 9) is now draft-ietf-core-groupcomm-bis-00

[1] https://mailarchive.ietf.org/arch/msg/core/fme0kaeiiroi6ETKxD3yoD_MiyE/

[2] https://mailarchive.ietf.org/arch/msg/core/TgmEmwhDB2EokFkMCh8UWgOxO8E/

# Groupcomm-bis-00: content view

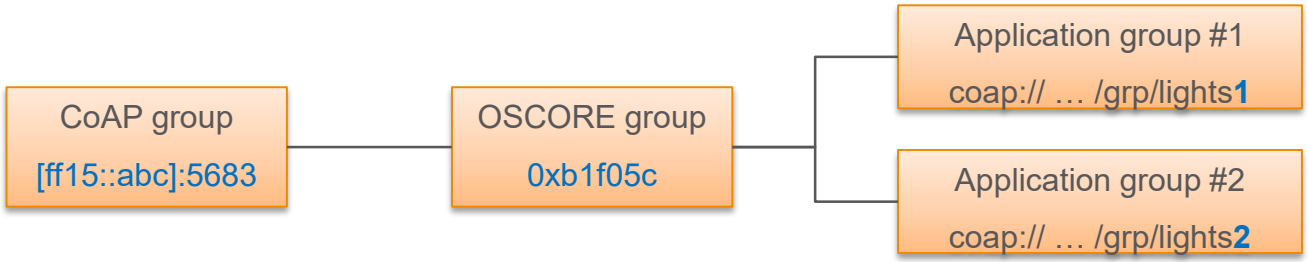
- Improved definition (2.1) of application/CoAP/security groups
  - including two new figures
- Added group discovery (2.2.3) with reference to RD.
- Security section on countering attacks (5.2.3) rewritten with more details
- Fixes & clarifications
  - improved description of RFCs that are obsoleted/updated
  - many others!

# Groupcomm-bis-00 “Group” concepts

› **Distinguish *types* of groups**
  – CoAP group: network level
  – OSCORE group (‘security group’)
  – Application group: application level

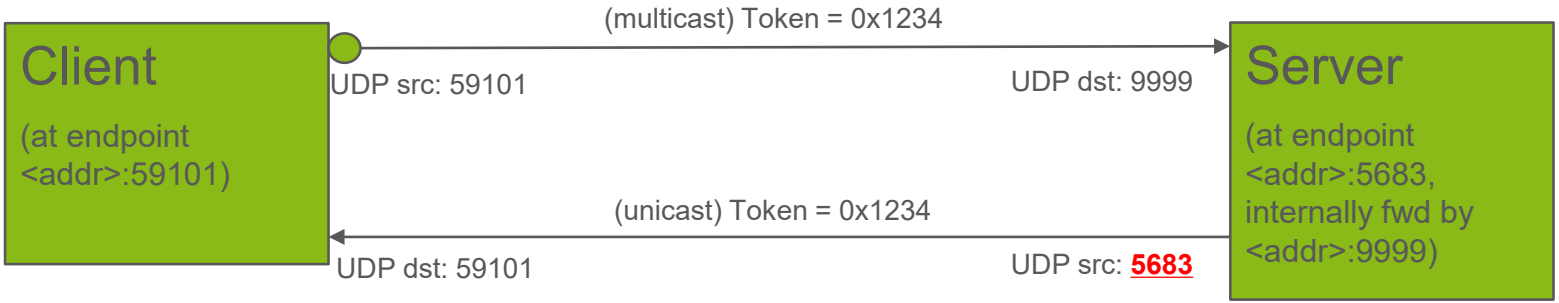
(identifiers for group type:)
→ multicast-address + port
→ Group name (invariant string)
→ <any application-specific ID>

› Example of group relations:



# Open Issues in Github / Gitlab

- See groupcomm-bis issues page and previous page
- #1 Clarify multicast endpoint concept and messaging model - UDP port may change
  - based on email thread [core] RFC 7252 - 8.2 - Multicast - Request / Response Layer, page 67, top



# Open Issues in Github / Gitlab

- See groupcomm-bis GitHub issues page and previous GitLab page

- #26 Section 2.1.2 - URI-Host for naming application groups

- #35 Consider if consistency requirement for "response suppression" should operate on Response Code class or not

# Next steps

- Work on issues in -00
- Process the latest review comments by Jim
- Test selected functions in CoAP implementations
  - E.g. “Observe + multicast” extension of RFC 7641 (first tests done successfully with Californium)

Thank you!

Comments/questions?

# Motivation (backup slide)

- RFC 7390 was published in 2014
  - CoAP functionalities available by then were covered
  - No group security solution was available to indicate
  - It is an Experimental document (started as Informational)
- What has changed?
  - More CoAP functionalities have been developed (Block-Wise, Observe)
  - RESTful interface for membership configuration is not really used
  - Group OSCORE provides group end-to-end security for CoAP
- Practical considerations
  - Group OSCORE clearly builds on RFC 7390 normatively
  - However, it can refer RFC 7390 only informationally

# Group OSCORE - Secure Group Communication for CoAP

draft-ietf-core-oscore-groupcomm-08

**Marco Tiloca**, RISE
Göran Selander, Ericsson
Francesca Palombini, Ericsson
Jiye Park, Universität Duisburg-Essen

IETF CoRE WG, Virtual Interim, April 8th, 2020

# Selected updates from -06

- Comments and reviews from Jim and Christian – Thanks!
  - Addressed specific comments from IETF 106
  - Addressed Jim's review of -06 [1]
  - Addressed Jim's review of -07 [2] (some open points left)
  - Addressed Christian's review of -07 [3] (some open points left)

[1] https://mailarchive.ietf.org/arch/msg/core/UEXWZLXP6VnpykN-C7A-Z0qYWxY/
[2] https://mailarchive.ietf.org/arch/msg/core/GdqlGpoLBi-2Q61N_iQeqXC5UL4/
[3] https://mailarchive.ietf.org/arch/msg/core/-F9oo5Ilo6TuZHv-6-vVCpFTd5k/

# Selected updates from -06

- Message processing across group rekeying
  - Responses always protected with the latest keying material
  - A response may be processed with a different context than the request
  - Include server's 'Partial IV' and new 'kid_context'

- Support for Observe
  - Dedicated sections for requests and response processing
  - The client 'kid' from the original Observe request is stored for reference

- Using group keying material for unicast requests: NOT RECOMMENDED
  - An external adversary can redirect the request to the group or a different server
  - Bad especially for non-safe methods; impact on Echo option and Block-wise

# Three modes of operations

- Three different protecting modes
  - **Signature mode** – Main and usual mode
    - Encryption with group keying material; signature included
  - **Optimized/Hybrid mode** – Section 9
    - Request: encryption with group keying material; stripped MAC; signature included
    - Response (*): encryption with derived pairwise keying material; no signature
  - **Pairwise mode (*)** – Appendix G
    - Encryption with derived pairwise keying material; no signature

(*) Not for use cases with an intermediary that verifies signatures

# Pairwise keys

- Key derivation
  - Same construction from 3.2.1 of RFC 8613
  - **Pairwise key = HKDF(Sender/Recipient key, DH shared secret, info, L)**
    - Sender Key of the sender node, i.e. Recipient Key of the recipient side
    - Static-static DH shared secret, from one's private key and the other's public key
  - Compatible with ECDSA and EdDSA (with mapping to Montgomery coordinates)

- New Pairwise Flag bit in the OSCORE option
  - Set to 1 if the message is protected with pairwise keying material
    - Optimized/Hybrid mode – Responses only
    - Pairwise mode – Requests and responses

# Open points

- Sender Sequence Number (SSN). Reset after rekeying?
  - Reset (as in OSCORE)
    - Pro: maximum lifetime of SSN, at each key epoch
    - Con: observations have to terminate after rekeying.
  - Don't reset --- Default behavior, app policies may override
    - Pro: observations can continue throughout a rekeying
    - Con: non-maximum lifetime of SSN, at each key epoch

- Optimized/hybrid mode
  - Concerns from Jim and Christian
  - Move to an appendix, and only about the optimized request
  - Instead, move the pairwise mode up in the document body

# Open points

- Normative statements on the modes. Proposal:
  - Signature mode MUST be supported
  - Pairwise mode MAY be supported
    - MUST be supported if Echo and/or Block-wise is supported
  - Applications can protect a request in one mode, and responses in another mode

- (a) OSCORE; (b) Group OSCORE in pairwise mode. Difference for a node?
  - a) Multiple full context establishments, on the wire
  - b) 1 full context establishment on the wire, through the Group Manager
    - Derivations of Recipient Contexts happen locally and when needed
  - The difference is about key management.
  - Add considerations about this in the section on pairwise mode?

# Open points

- Use of the pairwise mode in the group
  - Signaled as a group policy?

- Does the pairwise flag bit have a more general applicability? (Christian)
  - Thought about it with Group OSCORE in mind. No further obvious meanings.

- Should we flip the value of the pairwise flag bit? (Christian)
  - 0: Group OSCORE pairwise mode; same for OSCORE
  - 1: Signature mode
  - Need to (easily) update implementations

# Open points

- Error handling on not supporting the pairwise mode
  - Not so much to do on the client
  - The server can respond with an error, possibly with diagnostic information
  - Issues with that?

- Group ID in all notifications following a rekeying (Jim)
  - The client has two observations with the server
    - One observations with CTX1, one observation with CTX2
  - The server uses the same ‘kid’ in both CTX1 and CTX2
  - Is this really an issue?
    - The two observations started with two different requests, with different tokens
    - Tokens are associated to security contexts

# Open points

- Appendix E.2 – “Baseline” synchronization of Client’s Sequence Number
  - First request to be accepted or not by the server? (Christian, Jim)
- For the pairwise mode, the client has to know
  - Address, ‘kid’, and public key of the server
  - Generic discovery mechanisms in Appendix G.1. Good enough?
- Silent servers supporting the pairwise mode
  - Need to have a public key and a ‘kid’ as its identifier
  - These silent-server-only provide a public key, and get a Sender ID. Issues with that?
- Remove IANA registries on signature params and key params
  - Point at the recently extended registries in *cose-rfc8152bis-algs-07*
- Considerations on what should be done after reboot. New Appendix?

# Next steps

- Close open points
  - From Jim's and Christian's review of -07
  - Other pending issues raised today
  - From Jim's review of -08 [1] – Thanks!

- Test message protection in pairwise mode

- Once done, move to WGLC ?

[1] https://mailarchive.ietf.org/arch/msg/core/kmh1KjqEsR156m7EZ4yawaJnaG8/

# Thank you!

# Comments/questions?

https://github.com/core-wg/oscore-groupcomm

# Discovery of OSCORE Groups with the CoRE Resource Directory

draft-tiloca-core-oscore-discovery-05

Marco Tiloca, RISE
**Christian Amsüss**
Peter van der Stok

IETF CoRE WG, Virtual Interim, April 8th, 2020

# Recap

- A newly deployed device:
  - May not know the OSCORE groups and their Group Manager (GM)
  - May have to wait GMs to be deployed or OSCORE groups to be created

- Use the CoRE Resource Directory (RD):
  - Discover an OSCORE group and retrieve information to join it
  - Practically, discover the links to join the OSCORE group at its GM
  - CoAP Observe supports early discovery and changes in group information

- Use resource lookup, to retrieve:
  - The name of the OSCORE group
  - A link to the resource at the GM for joining the group

# Updates from -04

- Addressed review from Jim – Thanks!
  - https://mailarchive.ietf.org/arch/msg/core/FoNCVZtIRzYhv4Imx6e87ZoFk0w/
  - Still one open point (later slide)

- Improved content organization
  - Registration of Group Manager endpoints
  - List and description of target attributes

- Registration of links to ACE Authorization Servers

- Added examples in CoRAL
  - Also asked by Jim

# Link to Authorization Server

- When registering an OSCORE group to the RD
  - Possible to register related link to an Authorization Server (AS)
  - The AS is associated to the GM of the OSCORE group
- The joining node is able to retrieve the link to the AS
  - Avoid a first unauthorized access to the GM at joining time

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</group-oscore/feedca570000>;ct=41;rt="core.osc.mbr";
                              sec-gp="feedca570000";app-gp="group1";
                              cs_alg="-8";cs_alg_crv="6";
                              cs_key_kty="1";cs_key_crv=6";
                              cs_kenc="1",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/group-oscore/feedca570000"

Response: RD -> GM

Res: 2.01 Created
Location-Path: /rd/4521
```

# From Jim's review

- An application group can use multiple OSCORE groups
  - E.g., one for administration and one for normal communication

- Clarified meaning and usage of 'sec-gp'
  - Stable, invariant and plane name of the OSCORE group
  - This also makes *draft-ace-key-groupcomm-oscore* an informative reference

- Algorithm/key related parameters
  - Improved name and definitions

# Examples in CoRAL

- Covered all the main examples
  - Registration, Update with re-registration, Lookup #1, Lookup #2
- Many things become easier
- Easier to specify the link to the AS
  - Easy to add information to such link
  - That link is not to be “navigated”. Ok?
- Currently as Appendix
  - Plan to move to the document body

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/res
  ?rt=core.osc.mbr&app-gp=group1
Accept: TBD123456 (application/coral+cbor)

Response: RD -> Joining node

Res: 2.05 Content
Content-Format: TBD123456 (application/coral+cbor)

Payload:
#using <http://coreapps.org/reef#>
#using <http://coreapps.org/core.rd#>

#base <coap://[2001:db8::ab]/>
rd-item </group-oscore/feedca570000> {
   rt "core.osc.mbr"
   sec-gp "feedca570000"
   app-gp "group1"
   cs_alg -8
   cs_alg_crv 6
   cs_key_kty 1
   cs_key_crv 6
   cs_kenc 1
   as-uri <coap://as.example.com/token>
}
```

# Open point – BACnet example

- Explicit registration of node's membership to application groups
  - Nodes don't need to know their application groups in advance
- Issues
  - This results in multiple endpoint registrations
  - This is not a native functionality of the RD
- This document itself does not need this feature
  - But, it seems common practice in some deployments
- Possible way forward
  - Remove the membership registration from the BACnet example
  - Define the membership registration in a separate short document

# Summary and next steps

- Addressed Jim's review; link to AS; examples in CoRAL
- Outcome from previous meetings
  - "Time to start reading it in order to decide for WGA" [1]
  - People volunteered to review: Jim (done); Carsten; Klaus; Bill [1]
  - "Reviewer volunteers are asked to provide reviews now" [2]
- Way forward
  - Close the open point on registration of node's membership (BACnet example)
  - CoRAL: move examples to the document body; translate the BACnet example
  - Process reviews as they come

[1] https://etherpad.ietf.org/p/notes-ietf-104-core?useMonospaceFont=true
[2] https://mailarchive.ietf.org/arch/msg/core/78LHFFyq9c1_t0-kAmuDKcTzc3c/

# Thank you!

# Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-oscore-discovery

# Backup

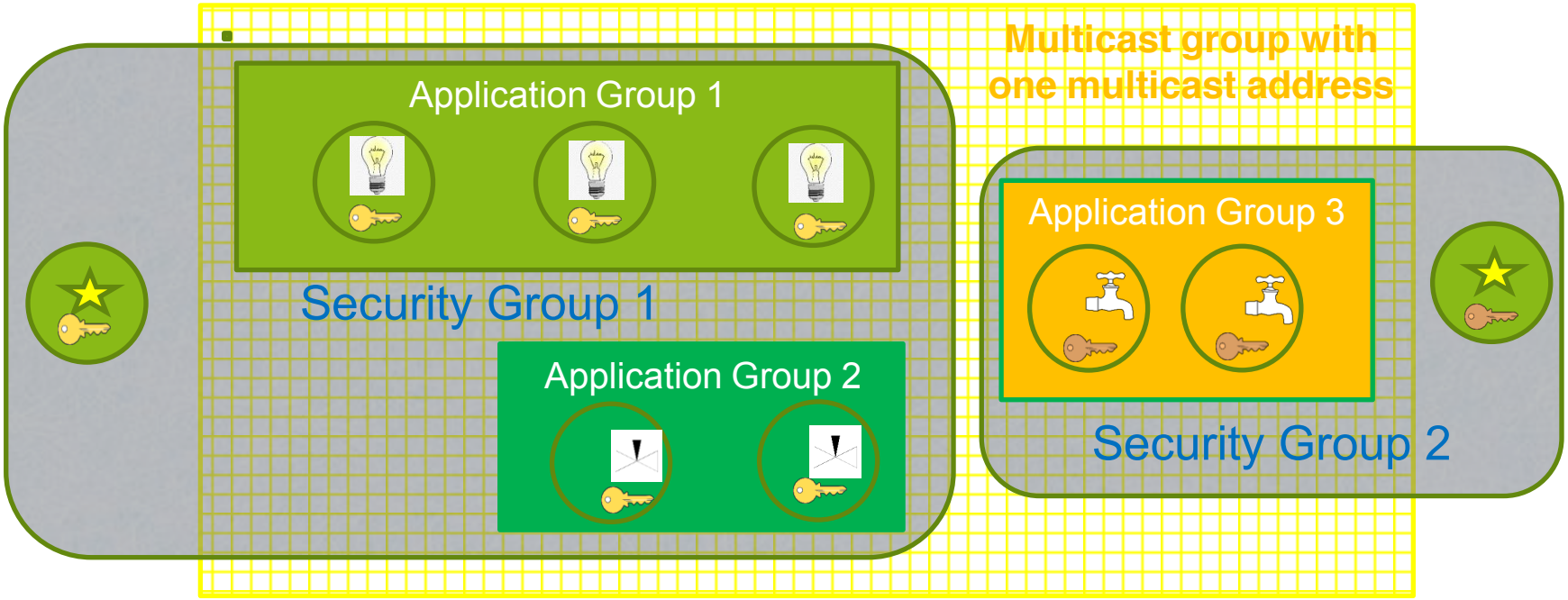
# Application/CoAP/Security Groups

- Application group
  - Defined in {RD} and reused as is
  - Set of CoAP endpoints sharing a pool of resources
  - Registered and looked up just as per Appendix A of {RD}

- CoAP/Multicast Group
  - Defined in draft-dijk-core-groupcomm-bis
  - Set of CoAP endpoints listening to the same IP multicast address
  - The IP multicast address is the ‘base’ address of the link to the application group

- OSCORE Security Group
  - Set of CoAP endpoints sharing a common Group OSCORE Security Context
  - A GM registers the group-membership resources for accessing its groups

# Application vs. Security Groups




Client of application group

Different key sets

Resources for given function

# Alg/key related parameters

- New optional parameters for a registered join resource
  - (*)(**) *cs_alg* : countersignature algorithm, e.g. “EdDSA”
  - (*) *cs_alg_crv* : countersignature curve (if applicable), e.g. “Ed25519”
  - (*) *cs_key_kty* : countersignature key type, e.g. “OKP”
  - (*) *cs_key_crv* : countersignature curve (if applicable), e.g. “Ed25519”
  - (*) *cs_kenc* : encoding of public keys, e.g. “COSE_Key”
  - (**) *alg* : AEAD algorithm
  - (**) *hkdf* : HKDF algorithm

- Benefits for a joining node, when discovering the OSCORE group
  - (*) No need to ask the GM or to have a trial-and-error when joining the group
  - (**) Decide whether to join the group or not, based on supported the algorithms

# Registration

- The GM registers itself with the RD
  - MUST include all its join resources, with their link attributes
  - New ‘rt’ value “core.osc.mbr” in the CoRE Parameters registry

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</group-oscore/feedca570000>;ct=41;rt="core.osc.mbr";
                             sec-gp="feedca570000";app-gp="group1";
                             cs_alg="-8";cs_alg_crv="6";
                             cs_key_kty="1";cs_key_crv=6";
                             cs_kenc="1",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/group-oscore/feedca570000"

Response: RD -> GM

Res: 2.01 Created
Location-Path: /rd/4521
```

# Discovery (1/2)

› The device performs a resource lookup at the RD
  – Known information: name of the Application Group, i.e. “group1”
  – Need to know: OSCORE Group Identifier; Join resource @ GM; Multicast IP address
  – ‘*app-gp*’ → Name of the Application Group, acting as tie parameter in the RD

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/res
  ?rt=core.osc.mbr&app-gp=group1

Response: RD -> Joining node


Res: 2.05 Content
Payload:
<coap://[2001:db8::ab]/group-oscore/feedca570000>;rt="core.osc.mbr";
    sec-gp="feedca570000";app-gp="group1";
    cs_alg="-8";cs_alg_crv="6";cs_key_kty="1";
    cs_key_crv=6";cs_kenc="1";anchor="coap://[2001:db8::ab]"
```

# Discovery (2/2)

- The device performs an endpoint lookup at the RD
  - Still need to know the Multicast IP address
  - 'ep'  // Name of the Application Group, value from '*app-gp*'
  - 'base'  // Multicast IP address used in the Application Group

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/ep
  ?et=core.rd-group&ep=group1

Response: RD -> Joining node

Res: 2.05 Content
Payload:
</rd/501>;ep="group1";et="core.rd-group";
          base="coap://[ff35:30:2001:db8::23]"
```

# Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-02

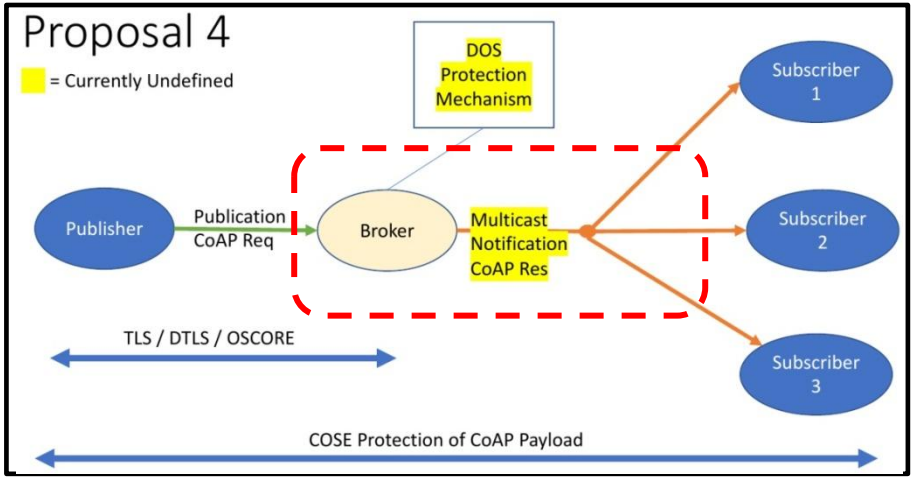
Marco Tiloca, RISE
Rikard Höglund, RISE
**Christian Amsüss**
Francesca Palombini, Ericsson

IETF CoRE WG, Virtual Interim, April 8th, 2020

# Recap

- Observe notifications as multicast responses
  - Many clients observe the same resource on a server S
  - Improved performance due to multicast delivery
  - Multicast responses are not defined yet. Token binding? Security?

- Practical use case
  - Pub-Sub scenario
  - Many clients subscribe to a same topic on the Broker
  - Better performance
  - Subscribers are clients only




From the Hallway Discussion @ IETF 104

# Contribution

- Define Observe notifications as multicast responses

- Management and enforcement of a common Token space
  - The Token space belongs to the group
  - The group entrusts the management to the server
  - All clients in a group observation use the same Token value

- Use of 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*

# Rationale

- The server can start a group observation for a resource, e.g. :
  1. With no observers yet, a traditional registration request comes from a first client
  2. With many traditional observations, all clients are shifted to a group observation

- 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 new/shifted clients an ***error response*** with:
  - Serialization of the phantom request
  - IP multicast address where notifications are sent to
  - current representation of the target resource

# Updates from -04

- New section on congestion control
  - Requested by Carsten at IETF 106
  - Building on *core-groupcomm-bis* and RFC 7641
- Encoding of the informative error response
  - New content format *informative-response+cbor*
  - New registry for parameter of informative response
  - Separate registry for parameters of phantom request
- Parameter meaning
  - *src_addr*, *src_port*, *dst_addr*, *dst_port* : addressing information
  - *coap_msg* : serialization of the phantom request (i.e. UDP payload)
  - *notif_num* : latest used observe number, as baseline for the client
  - *res* , *res_ct*: current resource representation and its content-format

**Informative error response**

```
Payload: { ph_req : {
             src_addr : bstr(M_ADDR),
             src_port : 65500,
             dst_addr : bstr(SERVER_ADDR),
             dst_port : 7252,
             coap_msg : bstr(PH_REQ.CoAP)
           },
           notif_num : 10,
           res : bstr("1234"),
           res_ct : 0
         }
```

# Updates from -04

- Appendix A - Alternative ways to retrieve a phantom request

```
Request:

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

Response:

    2.05 Content
    Content-Format: CoRAL

    rdf:type <http://example.org/pubsub/topic-list>
    topic    </ps/topics/1234> {
        dst_addr h"ff35003020010db8..1234"
        src_port 5683
        dst_addr h"20010db80100..0001"
        dst_port 5683
        coap_msg h"120100006464b431323334"
    }
```

- Pub-Sub
  - The phantom request is part of the topic metadata
  - A subscriber gets it already upon topic discovery
  - Early listening for multicast observations

- Sender introspection
  - Useful for debugging upon intercepting notifications
  - Query the server on a dedicated interface

```
Request:

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

Response:

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

    {'ph_req': {
        'dst_addr': h"ff35003020010db8..1234"
        'src_port': 5683
        'dst_addr': h"20010db80100..0001"
        'dst_port': 5683
        'coap_msg': h"120100006464b431323334"
    }}
```

# Updates from -04

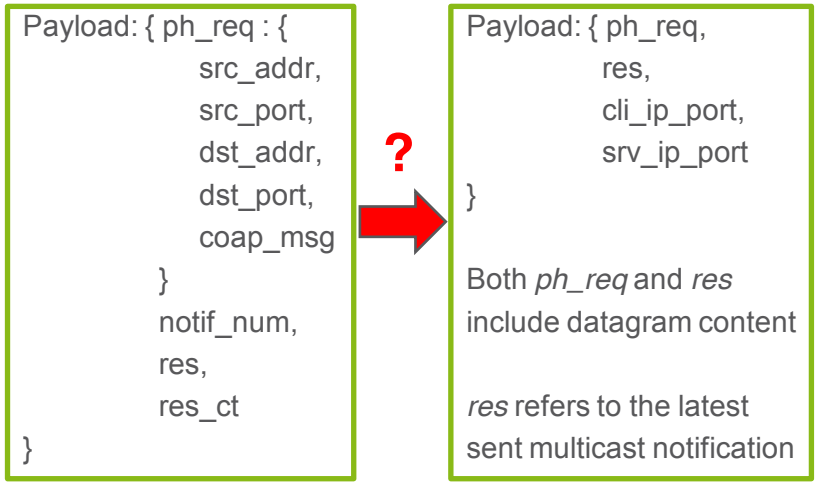
- Cancellation of group observation
  - The server sends to itself a phantom cancellation request
  - A multicast 5.03 response follows, with no payload
- When? Not enough clients are still active
  - Proposal: rough counting of alive clients, with a poll for interest
  - New CoAP options for successful multicast notifications
- Server current rough estimate: *n*
  - Expected confirmations *m* < *n*
  - Option value: *q* = ceil (*n* / *m*)
  - Each client picks a random *c* : [ 0, *q* )
  - If *c* == 0, the client sends a registration request (Non; with No-Response)
  - The server receives *r* of such requests, than *n* ← (*r* * *q*)

| No. | C | U | N | R | Name | Format | Len. | Default |
|---|---|---|---|---|---|---|---|---|
| TBD | | x | | | Multicast-Response-Feedback-Divider | uint | 0-8 B | (none) |

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

# Open points

- Informative error response in CoRAL
  - Early version already in Appendix A
- Considerations on the rough counting of alive clients
  - When stop waiting for confirmations? Leisure time + some transmit time …
  - Good practices and checks to be sure avoiding Smurf attacks
- Alternative encoding of the informative request
  - Now the info on the current resource is split
  - Serialize it as the phantom request in coap_msg?
  - Pro: use the native Observe numbers



# Summary

- Multicast notifications to all clients observing a resource
- Latest additions
  - Media type and encoding for the error response
  - Cancellation of group observation, based on rough counting of clients
  - Alternative ways to retrieve a phantom request
- Open points to address
  - Considerations and parameter tuning for the client rough counting
  - Encoding within the error response (full notification vs. resource representation)
  - Error response in CoRAL (already sketched in the Appendix)
  - Error response using the format from *core-coap-problem* ?
- Need for document reviews

# Thank you!

# Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-observe-responses-multicast

# Backup

# Assumptions

- Clients have previously discovered the resource to access
- The server knows the IP multicast address where to send notifications
- If Group OSCORE is used to secure multicast notifications
  - The server has previously joined the right OSCORE group
- The server provides the clients with other required information

# 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 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, with no reply right away

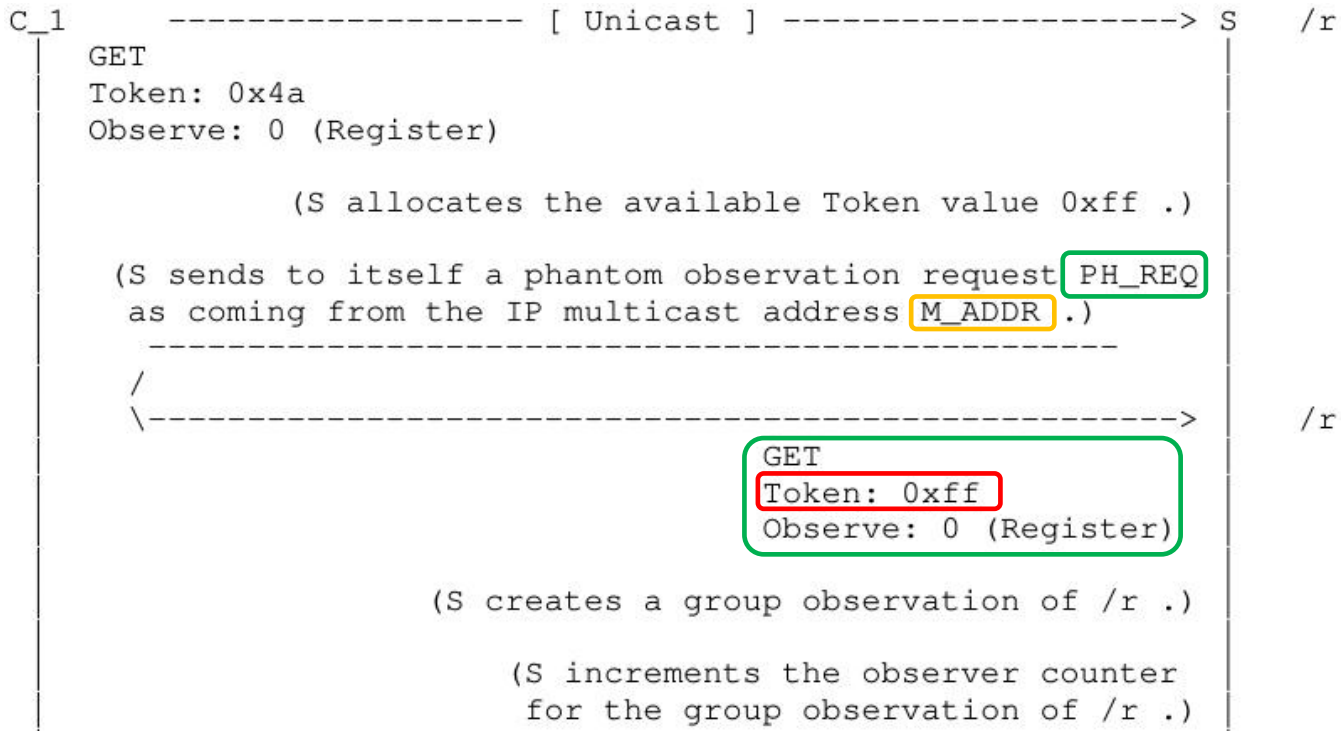
# Interaction with clients

- The server sends to new/shifted clients an ***error response*** with
  - '*ph_req*': byte serialization of the phantom request + Multicast IP addres + …
  - ' *res*': current representation of the target resource
  - '*notif_num*' and '*res_ct*': observe counter and content-format for the resource

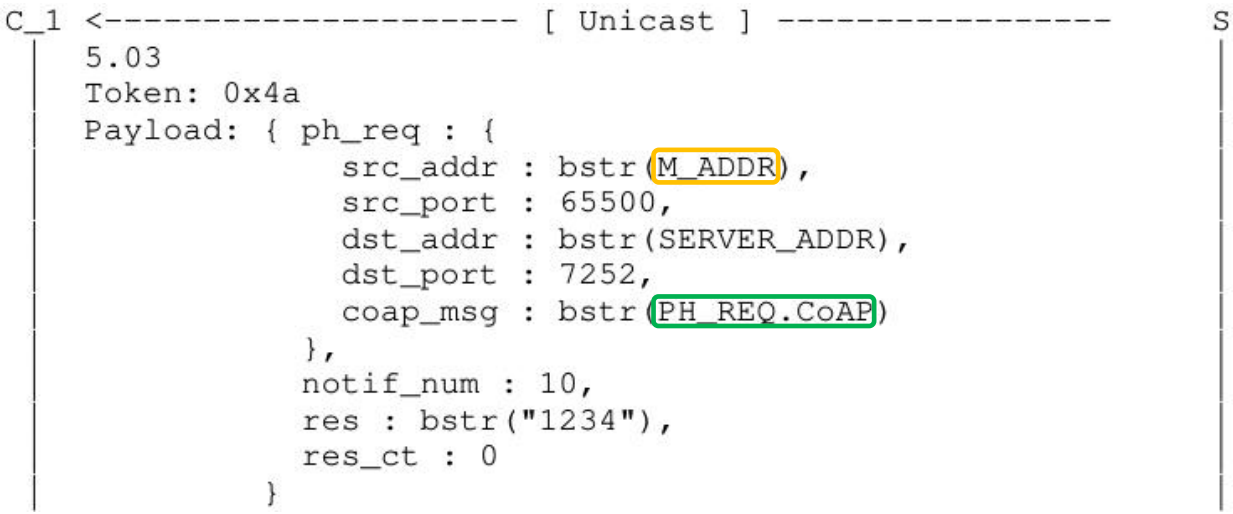
- When the value of the target resource changes
  - The server sends an Observe notification to the IP multicast address
  - The notification has the Token value T of the phantom request

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

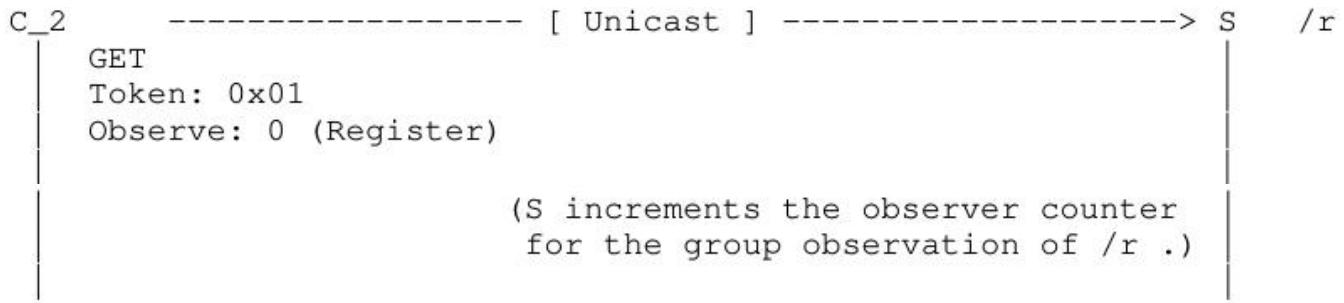
# C1 registration



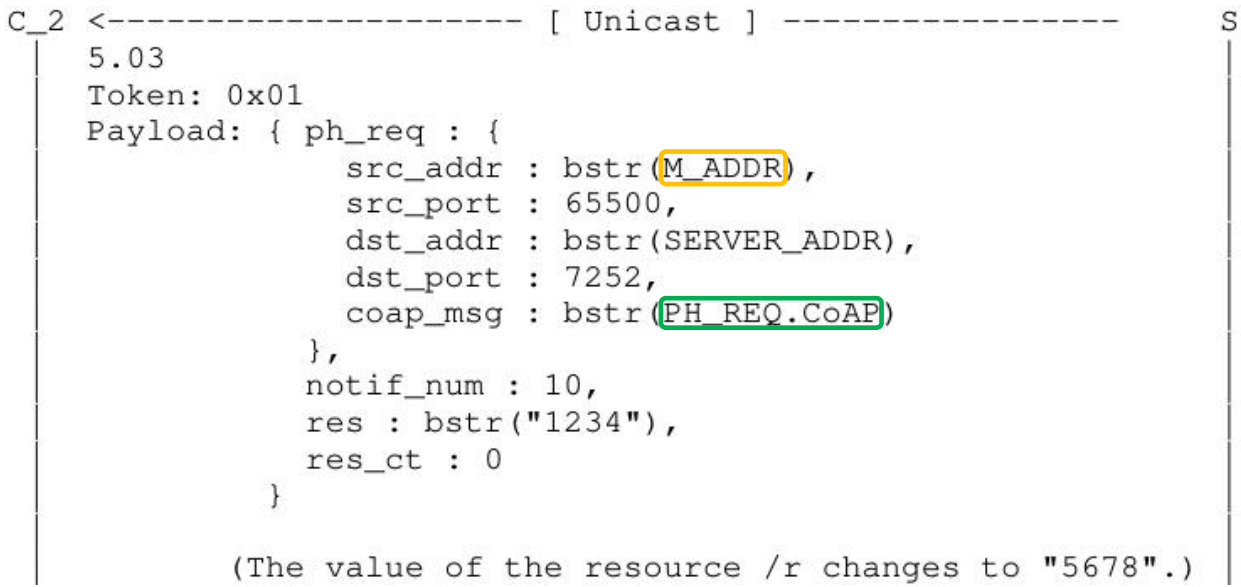
# C1 registration



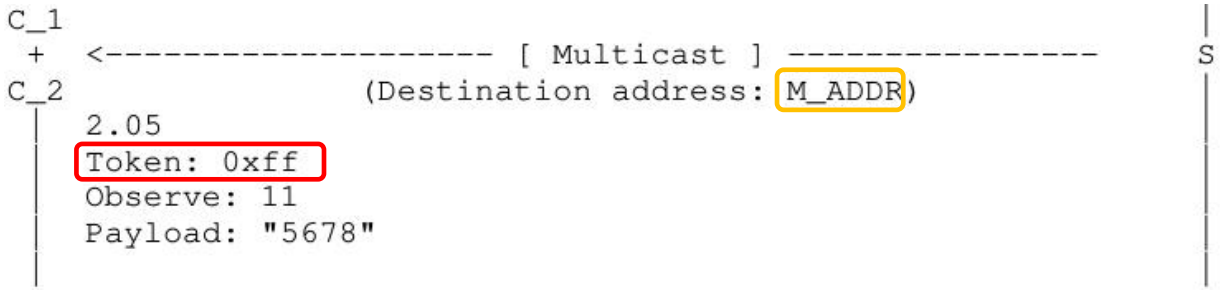
# C2 registration



# C2 registration



# Multicast notification




- 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
  - 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:
  - 'req_kid' = ***x***
  - 'req_piv' = ***y***

- The phantom request is still included in the informative response
  - Each client retrieves ***x*** and ***y*** from the OSCORE option

# Security with Group OSCORE

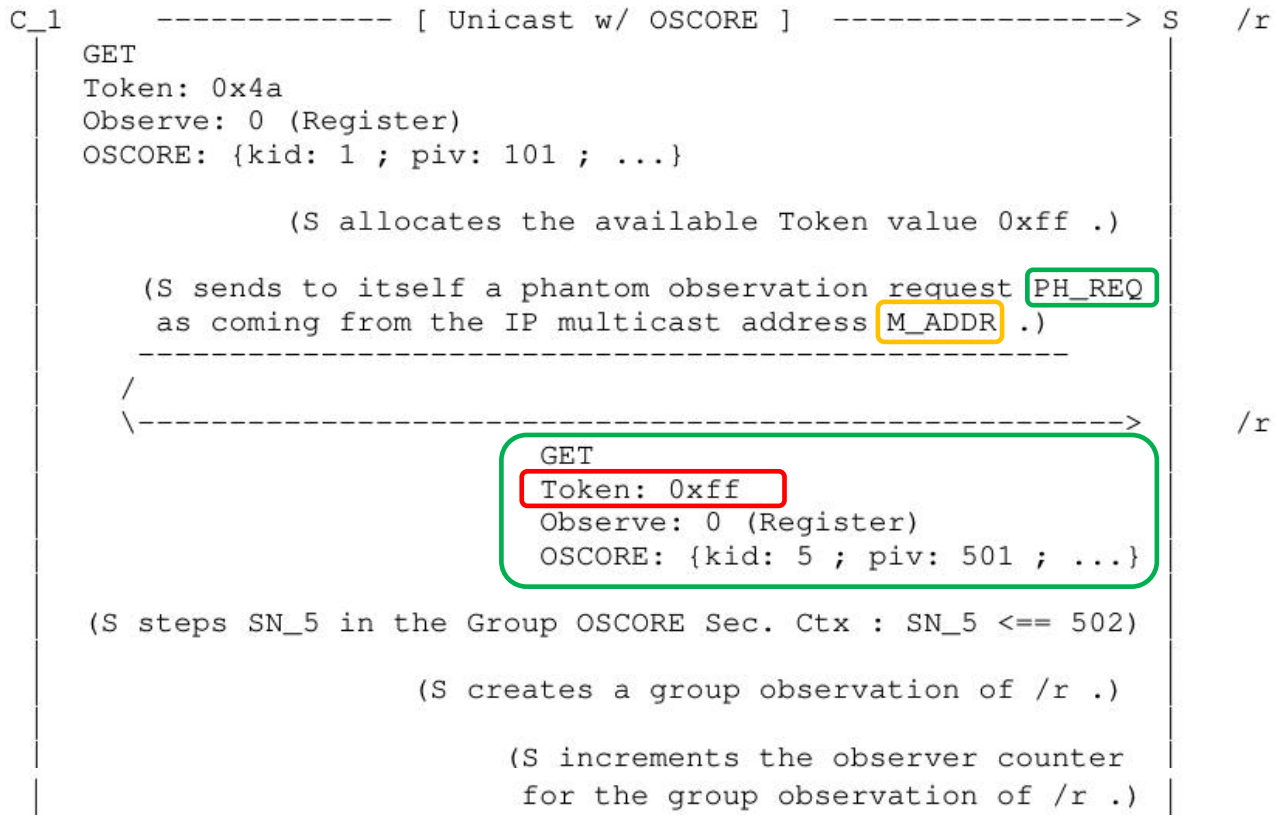
- In the error response, the server can ***optionally*** specify also:
  - '*join-uri*' : link to the Group Manager to join the OSCORE group
  - '*sec-gp*' : name of the OSCORE group
  - '*as-uri*' : link to the ACE Authorization Server associated to the Group Manager
  - '*cs-alg*' : countersignature algorithm
  - '*cs-crv*' : countersignature curve
  - '*cs-kty*' : countersignature key type
  - '*cs-kenc*' : countersignature key encoding
  - '*alg*' : AEAD algorithm
  - '*hkdf*' : HKDF algorithm

MUST

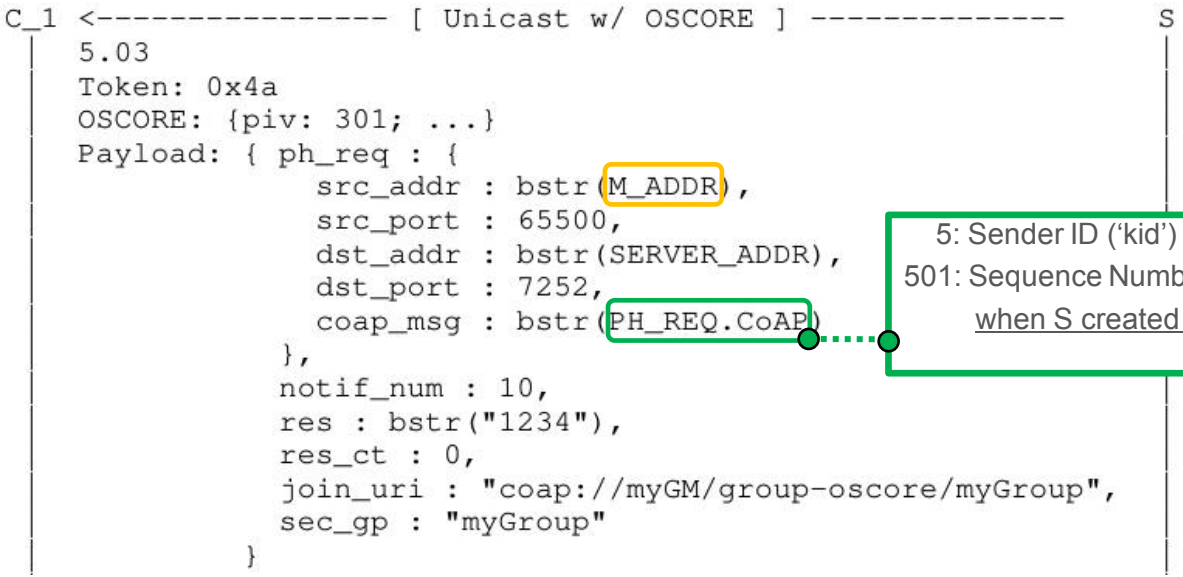
MAY

- Clients can still discover the OSCORE group through other means
  - E.g., using the CoRE Resource Directory, as in *draft-tiloca-core-oscore-discovery*

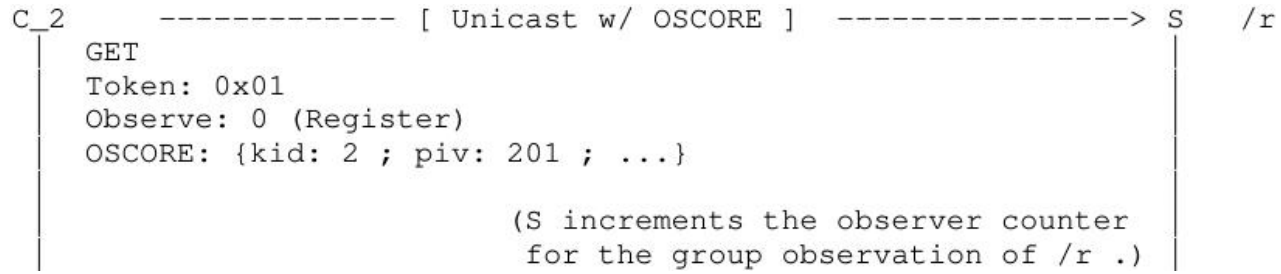
# C1 registration w/ security



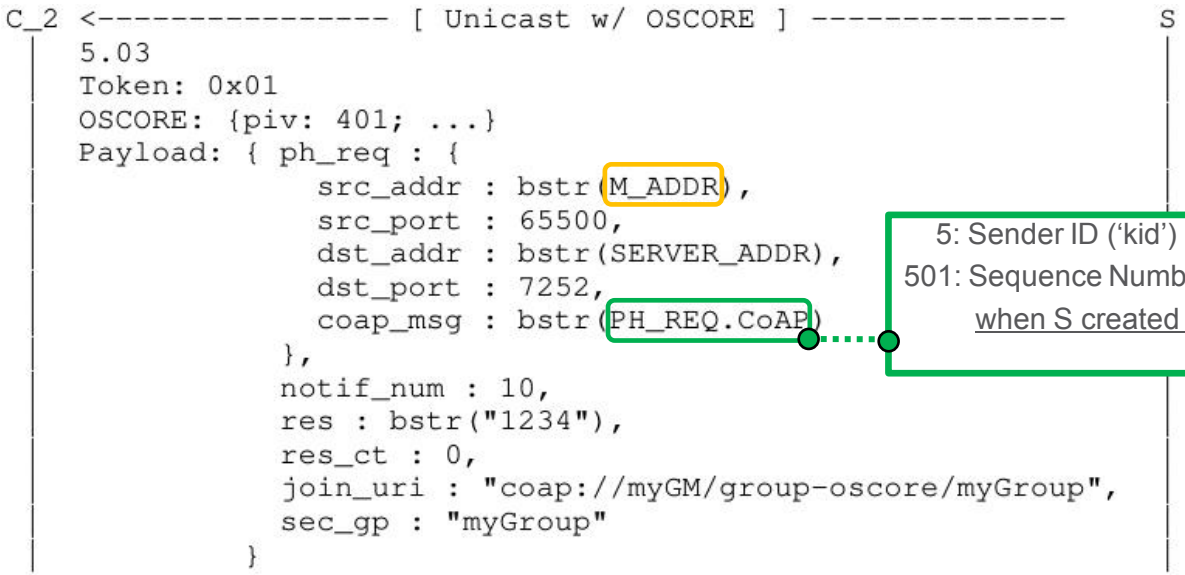
# C1 registration w/ security



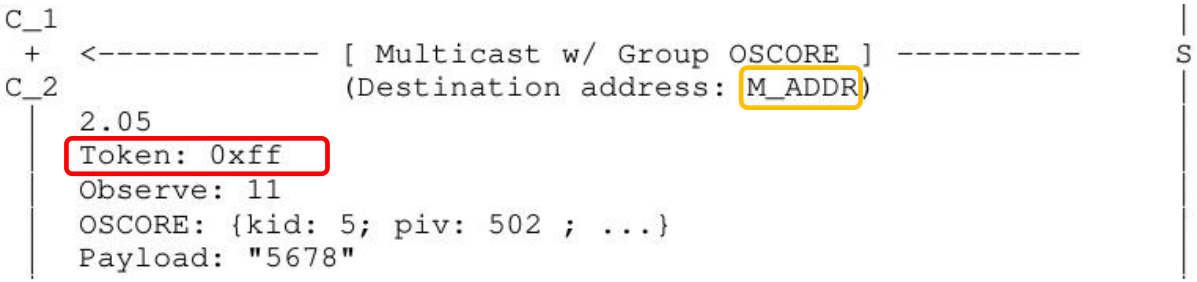
# C2 registration w/ security



# C2 registration w/ security



# Multicast notification w/ security




- When encrypting and signing the multicast notification:
  - The OSCORE *external_aad* has 'req_kid' = 5 and 'req_iv' = 501
  - 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

# Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-00

Marco Tiloca, RISE
**Esko Dijk, IoTconsultancy.nl**

IETF CoRE WG virtual interim, April 8th, 2020

# Motivation

- CoAP supports group communication over IP multicast
  - *draft-ietf-core-groupcomm-bis*

- The use of proxies introduces a number of issues
  - Clients to be whitelisted and authenticated on the proxy
  - The client may receive multiple responses to a single *unicast* request
  - The client may not be able to distinguish responses and origin servers
  - The proxy does not know when to stop handling responses

- Possible approaches for proxy to handle the responses
  - Individually forwarded back to the client
  - Forwarded back to the client as a single aggregated response

# Contribution

- Description of proxy operations for CoAP group communication
  - Addressed all issues in *draft-ietf-core-groupcomm-bis*

- Considered approach to handle responses:
  - Individually forwarded back to the client

- Assumptions
  - The proxy is explicitly configured to support group communication
  - Clients are whitelisted on the proxy, and identified by the proxy
  - Group OSCORE is used for secure group communication (end-to-end, client to server).

# Rationale

- Signaling protocol with two new CoAP options
  - Along the lines of Thomas' comments for *draft-dijk-core-groupcomm-bis*

- In the request addressed to the proxy, the client indicates:
  - To be interested in and capable of handling multiple responses
  - For how long the proxy should collect and forward back responses

- In a response to a group request, the server indicates its IP address
  - The client can distinguish the responses and the different servers
  - The client becomes able to (directly, or via proxy) contact the server individually via unicast

# Multicast-Signaling option

| No. | C | U | N | R | Name | Format | Length | Default |
|---|---|---|---|---|---|---|---|---|
| TBD1 | X | x | - | | Multicast-Signaling | uint | 1-5 B | (none) |

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

- Used only in requests
  - Presence: explicit claim of support and interest from the client
  - Value: indication to the proxy on how long to handle unicast responses
- Class I for OSCORE
  - Allows the proxy to see it but not to remove it

# Response-Forwarding option

| No. | C | U | N | R | Name | Format | Length | Default |
|---|---|---|---|---|---|---|---|---|
| TBD2 | X | x | - | | Response-Forwarding | (*) | 8-20 B | (none) |

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

- Used only in responses
  - Presence: allows the client to distinguish responses and originator servers
  - Value: IP address of the server, as a tagged CBOR byte string
- Class E for OSCORE

# Workflow: C -> P

- C prepares a request addressed to P
  - The group URI is included in the Proxi-Uri option or the URI-* options
- C chooses T seconds, as token retention time
  - T < Tr , with Tr = token reuse time
  - T considers processing at the proxy and involved RTTs
- C includes the Multicast-Signaling option, with value T′ < T
- C sends the request to P via unicast
  - C retains the token beyond the reception of a first matching response

# Workflow: P -> S

- P identifies C and verifies it is whitelisted
- P verifies the presence of the Multicast-Signaling option
  - P extracts the timeout value T′
- P forwards the request to the group of servers, over IP multicast
- P will handle responses for the following T′ seconds
  - Observe notifications are an exception – they are handled until the Observe client state is cleared.

# Workflow: S -> P

- S knows there's a client behind the proxy, by detecting the Multicast-Signaling Option.

- S includes the Response-Forwarding option in the response
  - The option value is the IP address of the server, as a tagged CBOR byte string

# Workflow: P -> C

- P forwards responses back to C, individually as they come
- P frees-up its token towards the group of servers after T′ seconds
  - Late responses > T' will not match and not be forwarded to C
  - Observe notifications are the exception
- C retrieves the Response-Forwarding option
  - C distinguishes different responses from different origin servers
  - C is able to later contact a server individually, either directly or indirectly
- C frees-up its token towards the proxy after T seconds
  - Again, Observe notifications are the exception

# Open points

- Mostly from Christian's comments – Thanks!
- Alternative design proposed – to consider
  - Proxy removes the Multicast-Signaling Option from request;
  - Proxy adds the Response-Forwarding Options and its IP address info to responses
  - No end-to-end security for the information in both Options
- If the proxy authenticates the client with a <C,P> OSCORE context …
  - We have a use case for "nested OSCORE"
  - Should we define it? Would this same document be appropriate?
- This document is general enough, as about "proxy operations"
  - Should it define also response aggregation as alternative approach?

# Summary

- Defined proxy operations for CoAP group communication
  - Embedded signaling protocol, using two new CoAP options
  - The proxy separately forwards back individual responses to the client for a defined time period T'
  - The client can distinguish the origin servers and corresponding responses
- Main next step: address Christian's comments and open points
- Need for comments and feedback

# Thank you!

# Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-groupcomm-proxy

# SenML

# SenML Data Value Content-Format Indication

draft-ietf-core-senml-data-ct-01

Ari Keränen, Carsten Bormann

IETF 107+, 2020-04-08, in the cloud

# Examples

```
{"n":"nfc-reader", "vd":"gmNmb28YKg", "ct":"60"}
```

```
{"n":"nfc-reader-42",
 "vd":"H4sIAA+dmFwAAzMx0jEZMAQALnH8Yn0AAAA",
 "ct":"text/csv@gzip"}
```

# Feature objective: extensibility

- `ct` is generally ignorable (like any new SenML field)
- But we would like to also have a “must understand” version, `ct_`

- Issue: Interaction between the two (`bct`, `bct_`) and resolved records
- Would prefer to have specific information (in record) override base
- But now, that happens only separately, within the thread for each field name!

# RFC 8428: “Must understand” and “_”?

- »Extensions that are **mandatory** to understand to correctly process the Pack MUST have a label name that ends with the "_" character.«
- »Applications MUST ignore any JSON **key-value pairs** that they do not understand unless the key ends with the "_" character, in which case an error MUST be generated.«
  (12.3.1 for senml+json, equivalent text for other representations)
- So a receiver is free to ignore a **key-value combination** if it doesn’t understand the key **or** if it doesn’t understand the combination
- Note that `foo` and `foo_` are different fields from a SenML perspective, except possibly by their semantic definition
  - convention: don’t define a `foo` and a `foo_` that are unrelated

# RFC 8428: ct, ct_, bct, bct_

- Resolving algorithm can be performed without understanding field semantics: no inter-field interaction
  - Fields do define how base value and given value for that field mix
  - »A future specification that defines new base fields needs to specify how the field is resolved.«
- Resolving is not influenced by unrelated fields (ct vs. ct_):
  It happens **separately** for ct and for ct_
- The rules applying to a record are applied **after** resolving

- But we need to look at examples having some of these four
  and see whether what we built makes sense

# Solution option #1

- Do not apply base value (bct or bct_) if a current value (ct or ct_) exists in the record
- Not supported by RFC 8428
  - Would require using new version/feature for SenML

# Solution option #2

- Future specification need to specify semantics of the "safe-to-ignore" and "must understand" versions of the same field in the same record
  - ct_ is the first registration of "must understand" fields
  - Can be handled as DE guidance and clarified in SenML-bis?
- Easy to avoid problem: don't mix the two variants in the Packs
  - but also need to enable combining packs easily
- For ct draft: if both exist in the same Record: ct_ overrides ct (i.e., ignore/remove "safe-to-ignore" version)
- Not perfect, but we don’t know better without new SenML version

# What we don’t like about solution #2

- If a pack has a bct_, you can no longer usefully use bct or ct *from that position on*
- That is a limitation, but it doesn’t detract from other useful combinations
- Workaround: Instead of using bct_, use ct_ once to check the must-understand feature; can use bct then

- To do: designated expert to write a wiki page explaining all this

# Mixing b and _ fields: what are the resolution rules?

1)

```
[
 {"bfoo_":42, "n":"t1", "v":1},
 {            "n":"t2", "v":2},
 {"foo": 1,   "n":"t3", "v":3}
]
```

2)

```
[
 {"bfoo_":42, "n":"t1", "v":1},
 {            "n":"t2", "v":2},
 {"foo_": 1,  "n":"t3", "v":3}
]
```

3)

```
[
 {"bfoo":42, "n":"t1", "v":1},
 {           "n":"t2", "v":2},
 {"foo_": 1, "n":"t3", "v":3}
]
```

4)

```
[
 {"bfoo":42, "n":"t1", "v":1},
 {           "n":"t2", "v":2},
 {"foo": 1,  "n":"t3", "v":3}
]
```

# Mixing b and _ fields: resolved

1)

```
[
 {"bfoo_":42, "n":"t1", "v":1},
 {            "n":"t2", "v":2},
 {"foo": 1,   "n":"t3", "v":3}
]
```

```
[
 {"foo_":42, "n":"t1", "v":1},
 {"foo_":42, "n":"t2", "v":2},
 {"foo": 1, "foo_":42", "n":"t3", "v":3}
]
```

# Mixing b and _ fields: resolved

2)

```
[
 {"bfoo_":42, "n":"t1", "v":1},
 {            "n":"t2", "v":2},
 {"foo_": 1,  "n":"t3", "v":3}
]
```

```
[
 {"foo_":42, "n":"t1", "v":1},
 {"foo_":42, "n":"t2", "v":2},
 {"foo_": 1, "n":"t3", "v":3}
]
```

# Mixing b and _ fields: resolved

3)

```
[
 {"bfoo":42,  "n":"t1", "v":1},
 {            "n":"t2", "v":2},
 {"foo_": 1,  "n":"t3", "v":3}
]
```

```
[
 {"foo":42,  "n":"t1", "v":1},
 {"foo":42,  "n":"t2", "v":2},
 {"foo_": 1, "foo":42, "n":"t3", "v":3}
]
```

# Mixing b and _ fields: resolved

4)

```
[
 {"bfoo":42,  "n":"t1", "v":1},
 {            "n":"t2", "v":2},
 {"foo": 1,   "n":"t3", "v":3}
]
```

```
[
 {"foo":42,  "n":"t1", "v":1},
 {"foo":42,  "n":"t2", "v":2},
 {"foo": 1,  "n":"t3", "v":3}
]
```

# SenML Features and Versions

draft-bormann-core-senml-versions-01

Carsten Bormann

IETF 107+, 2020-04-08, in the cloud

# RFC 8428, SenML: Version 10

- RFC 8428 SenML evolution path: allows for version upgrade
- Default version: 10 (accounting for previous development versions)
- Can set higher:  {"bver":11, "v":4711}
- Semantics to be defined by RFC updating RFC 8428

# Objective: extensibility

- Over time, new specifications will add features to SenML
- Version number is a unitary declaration:
  implementation of certain features is needed by the receiver to process SenML pack

- Version number N+1 includes all features of version number N (total order)
  - Except for features that are **deprecated**

# Version numbers are stupid

- Well, they work well for document revisions and software releases
- Not so great for protocols and other interface specifications
- Long discussion in T2TRG:
  Version numbers force creating a total order on a set of new features
- Better: declare individual features
  - Could do with must-understand fields: `bfeature1_: true`
  - But maybe can leverage the version number?

# Proposal: interpret version number as bits

- A number can be used as a bit array
- Version 10 = $1010_2$, i.e. features 1 and 3 ($2^1 + 2^3 = 10$)
- Add bits for additional features
- Proposed feature 4: use of Secondary Units ($2^4 = 16$)
  Version number with that additional feature would thus be 26
- Feature code can go up to 52 (53-bit integers in JSON):
  48 remaining now (after secondary unfits)

# 53: wasn’t that an evil number?

- Yes.
- But it could be all we need:
  - As the number of features that can be registered has a hard limit (48 codes left at the time of writing), the designated expert is specifically instructed to maintain a frugal regime of code point allocation, keeping code points available for SenML Features that are likely to be useful for non-trivial subsets of the SenML ecosystem.
  - Quantitatively, the expert could for instance steer the allocation to not allocate more than 10 % of the remaining set per year.

# draft-bormann-core-senml-versions-01

- Defines the feature system:
  New Registry under the SenML registry
  Reserving feature code 0..3 for “10 = $1010_2$”
  Specification required, frugality mandate to designated expert
- **Updates** the RFC 8428 version number to use that system
- Registers feature code 4: Use of secondary units

- Useful?
- Ready for working group adoption?

# A link relation type for disclosing implementation information

draft-bormann-t2trg-rel-impl-01

Carsten Bormann

IETF 107+, 2020-04-08, in the cloud

# Implementation information helps debugging

- HTTP has `Server:`, `User-Agent:`
- CoAP: Not great to send this with every request/response
- Server side: Make information **discoverable**
- **/.well-known/core**: natural place
- Don’t put the actual information there, but a link
- Need a link relation type then

# draft-bormann-t2trg-rel-impl-01

- Defines link relation type `impl-info` for linking to implementation information
- Does **not** define media types this could point to
  - We could do that later
  - HTML is a great media type, too
- Discusses security considerations of disclosing implementation information
- Briefly touches on DDoS mitigation

# I’m done here, but:

- There is a controversial proposal known as `security.txt` draft-foudil-securitytxt-09, ostensibly for vuln reporting (and hiring)
- Shouldn’t rel-impl do something similar?
- No:
  - security.txt is for websites, not for devices
  - Pet vs. cattle
  - Implementation information can be set by manufacturer; security.txt merges this with PIL (purpose in life), operator contact, policy, … Not clear this (or link to this) is best kept in device.
- Yes: ? Discuss.

# Thank you!
# Comments/questions?