Agenda

- Document status
  - OSCORE, RD, dynlink
  - hop-limit, stateless, ERT, multipart-core
  - what else?
- SenML: units, data-ct, fetch/patch
- Pubsub empty

Document status

hop-limit: re-review of -03 update needed, go to WGLC
stateless: -01 to be submitted March 11
ERT: re-review of github version needed, ➔ -04

SenML units (draft-bormann-senml-more-units-00**)**

`B` (byte), `VA` (volt-ampere), `var` (volt-ampere-reactive)
`J/m` (Joule per meter, covering Wh/km etc.)

Should we go ahead with registering these now?

More might come before this becomes an RFC

(Does this need to be an RFC? What status?)

SenML data-ct (draft-keranen-core-senml-data-ct-00**)**

Currently, ct is an unsigned integer (content-format number)

- Should content-types (media types, with parameters) be allowed?
  - This would allow a string-values alternative
- content-codings? Separate string, separate field.
  - Maybe also for Content-Format values?

SenML fetch/patch (draft-ietf-core-senml-etch-02.txt)

Basebase issue:

```
[{"bn":"/3303/0/", "n":"5700","v":22.4},
                  {"n":"5601","v":12.2},
                  {"n":"5602","v":34.2},
 {"bn":"/3303/1/", "n":"5700","v":18.2},
                  {"n":"5601","v":9.2},
                  {"n":"5602","v":27.2}]
```

"bn" here is meant to be relative to some basename

- "evaluated in the context of the target resource" — what?
- basename of the pack to be patched?

➔ which base? that can change…

PubSub empty

REST method GET

2.05: return content
4.xx, 5.xx: exception

```
public Employee getByName(String name) {
  int id = database.find(name);
  return new Employee(id);
}
```

E.g., 4.04 is treated as an exception

Adding "empty" (2.07)

```
public Employee getByName(String name) {
  int id = database.find(name);
  if (id == 0) {
    return null;
  }
  return new Employee(id);
}
```

Special success code: `null`

Why?

Because we want an empty GET to “succeed”

— so we can use observe semantics

At what cost?

Every CoAP GET must now have a null check
(check for 2.07)

The billion dollar mistake (very conservative estimate)

I call it my billion-dollar mistake. It was the invention of the null reference in 1965. At that time, I was designing the first comprehensive type system for references in an object oriented language (ALGOL W). My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.

— Tony Hoare

Programming language evolution

1970's to 1990's languages (C, C++, C#, Java):
null references

2010's languages (C# 2→8, Haskell, Swift, Rust):
make the alternative outcome part of the return type!

Haskell: `Maybe Employee`
C#: nullable value type `Employee?` (8: reference types)
Swift: optional type `Employee?`

Media types (content types): Our type system

Problem: there are no type derivations

If I have an `Employee` (ct=4711), I can't talk about an `Employee?` (ct=???), unless these were registered in pairs

pubsub is introduced after the fact, we don't have the nullable alternatives

`application/multipart-core` is a catch-all for these types

loses part of Accept semantics (except: expressing ct-agility)

Practical issues

- Ease of Wrap/Unwrap
  - Wrap = add CBOR prefix
    - Need to encode length of embedded object
  - Unwrap = remove CBOR prefix
    - Need to decode (at least: skip) that length
- Could use a simpler encoding just for Maybe
  - E.g., 1 CBOR + blob

The billion $ question:
Are the practical issues bad enough
to make us redo the billion $ mistake?

Neither of these options are a no-brainer
(or really satisfying)

If both alternatives hurt, what is the way forward?

— Outlawing empty topics? ➔ (Create on publish)
— Contents don't expire any more?
(Or define tombstone with time for expiry)
— Tombstone = Like a last wish
— SenML: Empty pack; string formats: empty string,...
(needs to be defined by the application)

Alternative #4?

Allow Accept to represent a union type

That union type does not need to be reified as such (each GET would result in one of the elements of the union)

Client declares that it wouldn't be surprised with either result

Update observe to allow multiple content-formats if "Accept"ed by the client

Keep it simple

Don't provide for expiry

Implement ephemeral messages at the application level (use SenML, or multipart/core, ...)

➔ no concerns right now