

# SenML Features and Versions

Draft-ietf-core-senml-versions-01

Carsten Bormann

IETF 109, 2020-11-20, in the cloud

# Summary of IETF108 slides

RFC 8428, SenML: Version 10

Objective: extensibility

Version numbers are stupid

Proposal: interpret version number as bits

53: wasn't that an evil number?

draft-ietf-core-senml-versions-00

Next steps

# Next steps

IETF108

- Need more reviews!  
This is just about the interpretation for one field...
- Proposal: Process these reviews, check if we are done, WGLC
- IETF108 notes:  
»**People who will review: Bill, Jaime**«
- —01 makes minor edits (math presentation),  
updates RFC 8798 reference

IETF109

- **So, how about some reviews?**
- **And, how about implementations?**

Backup slides

# 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 units)



# 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-ietf-core-senml-versions-00

- Defines the feature system:
  - New Registry under the SenML registry
  - Reserving feature code 0..3 for “10 = 1010<sub>2</sub>”
  - 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
- Now WG draft, submitted 2020-05-13
  - Referenced from RFC 8798 (senml-more-units)
- No technical changes from 2020-03-06 draft-bormann-core-senml-versions-01