Sharing data models between SenML and CORECONF

draft-gudi-t2trg-senml-as-coreconf-00

7th November 2024

Manoj Gudi
(manoj.gudi@imt-atlantique.net)

What are we suggesting?

If SenML can be YANG'ified

then the its CORECONF-CBOR form
is similar to CBOR-SenML form*

because structurally
SenML ~ CORECONF

*Conditions Apply

Quick Review

- YANG: YANG Data modeling language (RFC 7950)
- SenML: Sensor Measurement Lists (RFC 8428)
- CBOR: Concise Binary Object Representation (RFC 8949)
- CORECONF: Coap Management Interface (draft-ietf-core-comi-18)
- YANG SIDs: YANG Schema Item iDentifiers (RFC 9595)

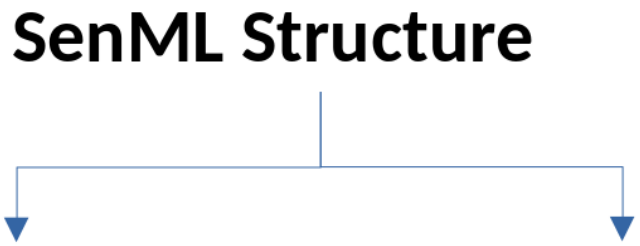
Why are we suggesting?

- Need to reduce payload-size and well-defined data models for the payload
- SenML CDDL may not be enough. YANG DM helps!
- CORECONF format maybe more efficient
- YANG → CORECONF → CBOR
- Leverage:
  - Existing YANG models (useful for telemetry data models)
  - Standardize Identifiers in SenML (such as measurement units)
  - Software tools around YANG (yanglint, pyang, pycoreconf, ccoreconf)

How does it work?

Write YANG Model for SenML

SenML → YANG: Overview



bn, bt, bv, bu ..

{ “bu” : “Celsius” }

n, t, u, v* ..

{ “n” : “temperature”, “v” : 25 }

```
leaf bu {
    type string;
    must "(. = 'm' or .= 'kg' or .. )";
}
```

```
choice valueleaf{
  leaf v{
    type Number;
  }
  leaf vs {
    type string;
  }
    ...
}
```

SenML → YANG: Tree

$ pyang -f tree senml.yang →

```
module: senml
  +--rw e* [t]
     +--rw bn?          string
     +--rw bt?          Number
     +--rw bu?          string
     +--rw bv?          Number
     +--rw bs?          Number
     +--rw bver?        Integer
     +--rw n?           string
     +--rw u?           string
     +--rw (valueleaf)?
     |  +--:(v)
     |  |  +--rw v?     Number
     |  +--:(vs)
     |  |  +--rw vs?    string
     |  +--:(vb)
     |  |  +--rw vb?    boolean
     |  +--:(vd)
     |     +--rw vd?    string
     +--rw s?           Number
     +--rw t            Number
     +--rw ut?          Number
```

SenML Labels

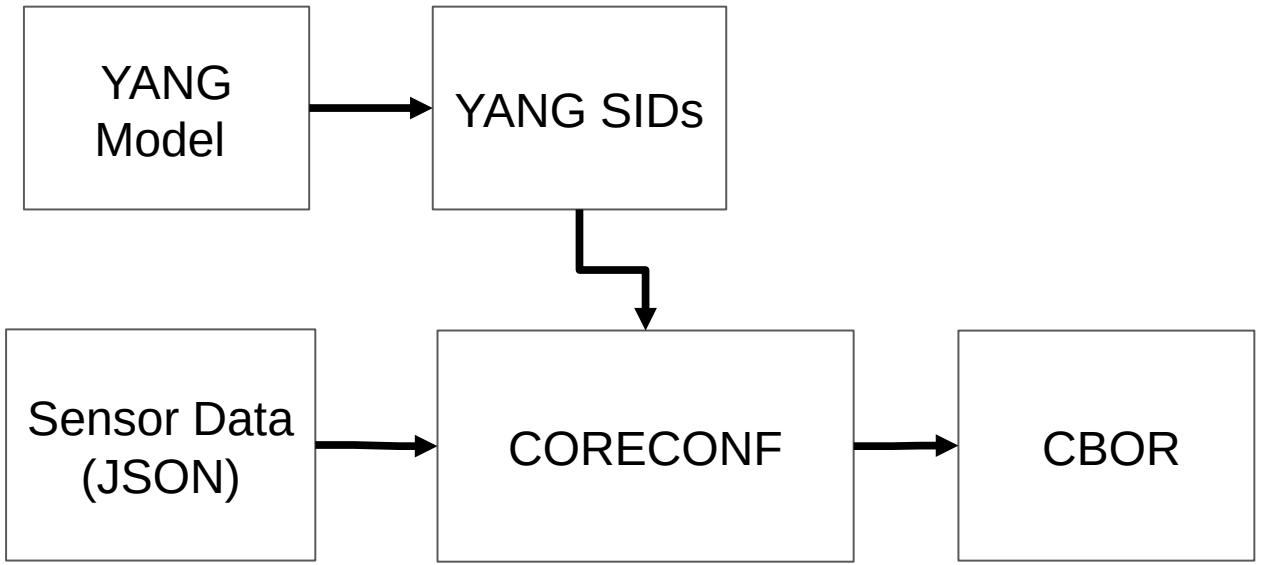
Name	Label	CBOR Label	JSON Type	XML Type
Base Name	bn	-2	String	string
Base Time	bt	-3	Number	double
Base Unit	bu	-4	String	string
Base Value	bv	-5	Number	double
Base Sum	bs	-6	Number	double
Base Version	bver	-1	Number	int
Name	n	0	String	string
Unit	u	1	String	string
Value	v	2	Number	double
String Value	vs	3	String	string
Boolean Value	vb	4	Boolean	boolean
Data Value	vd	8	String (*)	string (*)
Sum	s	5	Number	double
Time	t	6	Number	double
Update Time	ut	7	Number	double

Table 1: SenML Labels

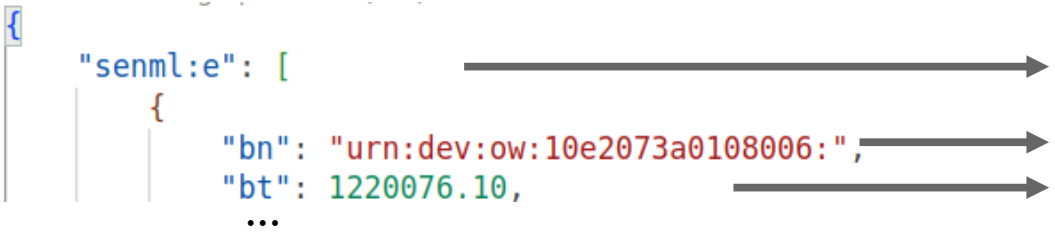
As defined in RFC 8428

CBOR Label for n is 0

CORECONF Overview



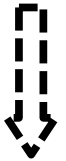
CORECONF Transformation



1. JSON

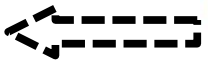
YANG Identifiers	SIDs
senml:e	60006
senml:e / bn	60007
senml:e / bt	60008
...	

2. Create SIDs



```
60006: [{
    60007: "urn:dev:ow:10e2073a0108006:",
    60008: 1220076.1,
    ...
```

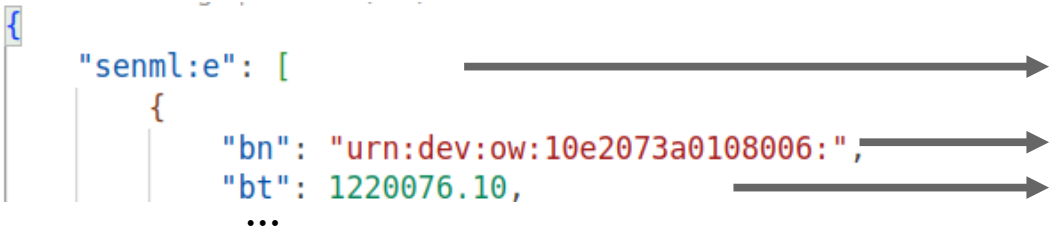
3. Substitute SIDs



```
60006: [{
    1: "urn:dev:ow:10e2073a0108006:",
    2: 1220076.1,
    ...
```

4. Delta Encoding CORECONF

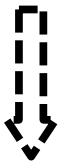
SenML CORECONF



1. JSON

YANG Identifiers	SIDs
senml:e	60006
senml:e / bn	60004
senml:e / bt	60003
...	

2. Modified SIDs



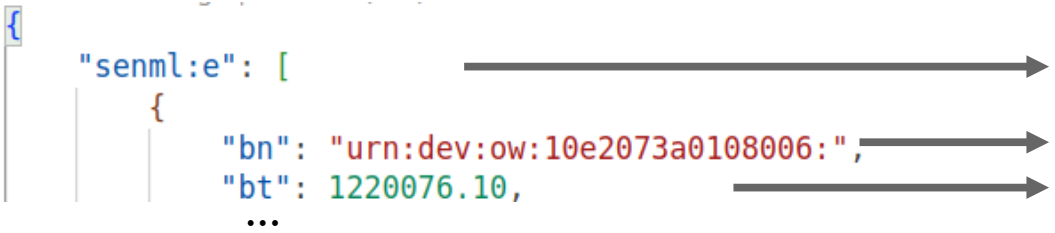
```
60006: [{
    -2: "urn:dev:ow:10e2073a0108006:",
    -3: 1220076.1,
    ...
```

4. Deltas Matching SenML Labels

```
60006: [{
    60004: "urn:dev:ow:10e2073a0108006:",
    60003: 1220076.1,
    ...
```

3. Substitute SIDs

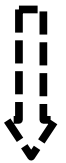
SenML CORECONF



1. JSON

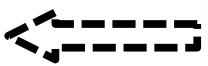
YANG Identifiers	SIDs
senml:e	60006
senml:e / bn	60004
senml:e / bt	60003
...	

2. Modified SIDs



```
60006: [{
    -2: "urn:dev:ow:10e2073a0108006:",
    -3: 1220076.1,
    ...
```

4. Deltas Matching SenML Labels



```
60006: [{
    60004: "urn:dev:ow:10e2073a0108006:",
    60003: 1220076.1,
    ...
```

3. Substitute SIDs

Remarks

The CBOR (hex) looks:
A1 19 EA 66 82 A6 21 ..

Module Identifier 4 bytes CBOR conforming to SenML-CBOR

First 4 bytes identifies the module

Subsequent CBOR conforms to SenML-CBOR and can be easily parsed

Caveats

- Hand modifying the SIDs
- Uniqueness of SIDs broken to get delta of 0

  (to match SenML Label of n = 0)
- Improve SenML-YANG Model to:
  - Add more constraints on units RFC 8798
  - Make leaf (v/vs/vb/vd) optional if Sum (s) is present

Opportunities

- Proposed YANG model is NOT the most optimal
- Possible to have non-zero CBOR labels for SenML?
- With an improved SenML YANG model, easily understand SenML data in your NETCONF-YANG devices
- Units can be modeled as YANG *identityrefs*?

Let’s try online!

coreconf.manojgudi.com

Transform To CORECONF

```
{
  "assignment-range": [
    {
      "entry-point": 60000,
      "size": 100
    }
  ],
  "module-name": "senml",
  "module-revision": "unknown",
  "item": [
    {
      "namespace": "module",
      "identifier": "senml",
      "status": "unstable",
      "sid": 60016
    },
    {
      "namespace": "data",
      "identifier": "/senml:e",
      "status": "unstable",
      "sid": 60006
    },
    {
      "namespace": "data",
      "identifier": "/senml:e/bn",
      "status": "unstable",
      "sid": 60004,
      "type": "string"
    },
```

```
{
    "e": [
        {
            "bn": "urn:dev:ow:10e2073a0108006:",
            "bt": 1220076.10,
            "n": "temperature",
            "u": "Cel",
            "v": 23.1,
            "t" : 1
        },
        {
            "n": "humidity",
            "u": "%RH",
            "v": 67.3,
            "t": 2
        }
    ]
}
```

Transform

SenML Example

CORECONF representation:

```
{60006: [{-3: 1220076.1,
          -2: 'urn:dev:ow:10e2073a0108006:',
          0: 'temperature',
          1: 'Cel',
          2: 23.1,
          6: 1},
         {0: 'humidity',
          1: '%RH',
          2: 67.3,
          6: 2}]}
```

CBOR Hex:

```
a119ea6682a621781b75726e3a6465763a6f773a313065323037336130313038303036 3a22fb41329dec1999999a006b74656d7065726174757265016
343656c02fb403719999999999a0601a4006868756d6964697479016325524802fb4050d333333333330602
```

JSON size: **144 B** | CORECONF size: **104 B** | Compressed: **≈ 27.78%**

Appendix

SenML → YANG: User Defined Types

- Create common types such as Number, Integer (JSON encoding)
- Number user-defined type to arbitrarily have 5 fraction digits for trade-off of precision and range [−92233720368547.75808, 92233720368547.75807]

```
typedef Number {
  type decimal64{
    fraction-digits "5";
  }
}
```

- Integer type made implementation agnostic using union type:

```
typedef Integer {
  type union{
    type int8;
    type int16;
    type int32;
    type int64;
  }
}
```

SenML → YANG: Base Fields

- Common-values or prefixes to be shared across multiple measurements/records
- base name (bn), base unit (bu), base time(bt), base unit(bu), base value(bv), base sum(bs), base version(bv)
- Optional, applies to all the records which contain it, and records after it (until there's a record with same base field)
- Example: *bn = "cpu/"* for *n = "temperature"* → *"cpu/temperature"*
- Most of these field types are primitive types (String, Numbers)
- Add YANG constraints on bu to ensure it can be of predefined unit (RFC 8798)

```
leaf bu {
    type string;
    must "(. = 'm' or .= 'kg' or .. )";
}
```

SenML → YANG: Regular Fields

- Fields containing actual unique sensor data, some of the fields are optional
- Similar to base fields except for the field Value (v) which can be either (for a measurement)
  - value (v) Number
  - String value (vs)
  - boolean value (vb)
  - data value (vd)
- Modeled as YANG choice type

n = 0 hack

```
{
    60006: [{
            -3: 1220076.1,
            -2: 'urn:dev:ow:10e2073a0108006:',
            "n": 'temperature',
            1: 'Cel',
            2: 23.1,
            6: 1
        },
```

Replace the δ for n from 0 to “n”

Consistent with YANG SID Spec. ✔
Breaks SenML CBOR Spec. x

Thank you for the hack Alexander Pelov!

Modifying YANG models

```
module mymodule{

   namespace
"urn:ietf:params:xml:ns:yang:mymodule";

   ...

   import senml{
       prefix sen;
}

   container myContainer{
       uses sen:senml-grouping;
          leaf newField{
               type int32;
}}}
```

```
augment "/sen:e"{
  uses sen:senml-grouping{
    refine bu{
       must "(. = 'newUnit1' or . = 'newUnit2')"{
         description
           "This rule should work the previous
            must keyword for bu to allow user
            to use two new units- newUnit1
            and newUnit2";

}}}}
```

FIN.