Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

C. Jennings
Cisco
Z. Shelby
ARM
J. Arkko
A. Keranen
Ericsson

October 19, 2015

Media Types for Sensor Markup Language (SENML) draft-jennings-core-senml-02

Abstract

This specification defines media types for representing simple sensor measurements and device parameters in the Sensor Markup Language (SenML). Representations are defined in JavaScript Object Notation (JSON), Concise Binary Object Representation (CBOR), eXtensible Markup Language (XML), and Efficient XML Interchange (EXI), which share the common SenML data model. A simple sensor, such as a temperature sensor, could use this media type in protocols such as HTTP or CoAP to transport the measurements of the sensor or to be configured.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Table of Contents

1. Overview

Connecting sensors to the internet is not new, and there have been many protocols designed to facilitate it. This specification defines new media types for carrying simple sensor information in a protocol such as HTTP or CoAP called the Sensor Markup Language (SenML). This format was designed so that processors with very limited capabilities could easily encode a sensor measurement into the media type, while at the same time a server parsing the data could relatively efficiently collect a large number of sensor measurements. There are many types of more complex measurements and measurements that this media type would not be suitable for. A decision was made not to carry most of the meta data about the sensor in this media type to help reduce the size of the data and improve efficiency in decoding. Instead meta-data about a sensor resource can be described out-of-band using the CoRE Link Format [RFC6690]. The markup language can be used for a variety of data flow models, most notably data feeds pushed from a sensor to a collector, and the web resource model where the sensor is requested as a resource representation (e.g., "GET /sensor/temperature").

SenML is defined by a data model for measurements and simple meta-data about measurements and devices. The data is structured as a single array that contains base value object(s) and array(s) of entries. Each entry is an object that has attributes such as a unique identifier for the sensor, the time the measurement was made, and the current value. Serializations for this data model are defined for JSON [RFC7159], CBOR [RFC7049], XML, and Efficient XML Interchange (EXI) [W3C.REC-exi-20110310].

For example, the following shows a measurement from a temperature gauge encoded in the JSON syntax.

```
[{}, [{ "n": "urn:dev:ow:10e2073a01080063", "v":23.5, "u":"Cel" }]]
```

In the example above, the first element of the root array is empty object since there are no base values. The second array inside the root array has a single measurement for a sensor named "urn:dev:ow:10e2073a01080063" with a temperature of 23.5 degrees Celsius.

2. Requirements and Design Goals

The design goal is to be able to send simple sensor measurements in small packets on mesh networks from large numbers of constrained devices. Keeping the total size of payload under 80 bytes makes this easy to use on a wireless mesh network. It is always difficult to define what small code is, but there is a desire to be able to

implement this in roughly 1 KB of flash on a 8 bit microprocessor. Experience with Google power meter and large scale deployments has indicated that the solution needs to support allowing multiple measurements to be batched into a single HTTP or CoAP request. This "batch" upload capability allows the server side to efficiently support a large number of devices. It also conveniently supports batch transfers from proxies and storage devices, even in situations where the sensor itself sends just a single data item at a time. The multiple measurements could be from multiple related sensors or from the same sensor but at different times.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4. Semantics

Each SenML representation carries a single array that represents a set of measurements and/or parameters. This array contains a base object with several optional attributes described below and a mandatory array of one or more entries.

Base Name: This is a string that is prepended to the names found in the entries. This attribute is optional.

Base Time: A base time that is added to the time found in an entry. This attribute is optional.

Base Units: A base unit that is assumed for all entries, unless otherwise indicated. This attribute is optional.

Version: Version number of media type format. This attribute is optional positive integer and defaults to 2 if not present.

The measurement or parameter entries array contains values for sensor measurements or other generic parameters, such as configuration parameters. There must be at least one entry in the array. This array is called simply "measurement array" in the following text.

Each array entry contains several attributes, some of which are optional and some of which are mandatory:

Name: Name of the sensor or parameter. When appended to the Base Name attribute, this must result in a globally unique identifier for the resource. The name is optional, if the Base Name is

present. If the name is missing, Base Name must uniquely identify the resource. This can be used to represent a large array of measurements from the same sensor without having to repeat its identifier on every measurement.

Units: Units for a measurement value. Optional.

Value Value of the entry. Optional if a Sum value is present, otherwise required. Values are represented using three basic data types, Floating point numbers ("v" field for "Value"), Booleans ("bv" for "Boolean Value") and Strings ("sv" for "String Value"). Exactly one of these three fields MUST appear.

Sum: Integrated sum of the values over time. Optional. This attribute is in the units specified in the Unit value multiplied by seconds.

Time: Time when value was recorded. Optional.

Update Time: A time in seconds that represents the maximum time before this sensor will provide an updated reading for a measurement. This can be used to detect the failure of sensors or communications path from the sensor. Optional.

The SenML format can be extended with further custom attributes placed in a base object, or in an entry. Extensions in a base object pertain to all entries following the base object, whereas extensions in an entry object only pertain to that entry.

Systems reading one of the objects MUST check for the Version attribute. If this value is a version number larger than the version which the system understands, the system SHOULD NOT use this object. This allows the version number to indicate that the object contains mandatory to understand attributes. New version numbers can only be defined in an RFC that updates this specification or it successors.

The Name value is concatenated to the Base Name value to get the name of the sensor. The resulting name needs to uniquely identify and differentiate the sensor from all others. If the object is a representation resulting from the request of a URI [RFC3986], then in the absence of the Base Name attribute, this URI is used as the default value of Base Name. Thus in this case the Name field needs to be unique for that URI, for example an index or subresource name of sensors handled by the URI.

Alternatively, for objects not related to a URI, a unique name is required. In any case, it is RECOMMENDED that the full names are represented as URIs or URNs [RFC2141]. One way to create a unique

name is to include a EUI-48 or EUI-64 identifier (a MAC address) or some other bit string that has guaranteed uniqueness (such as a 1-wire address) that is assigned to the device. Some of the examples in this draft use the device URN type as specified in [I-D.arkko-core-dev-urn]. UUIDs [RFC4122] are another way to generate a unique name.

The resulting concatenated name MUST consist only of characters out of the set "A" to "Z", "a" to "z", "0" to "9", "-", ":", ".", or "_" and it MUST start with a character out of the set "A" to "Z", "a" to "z", or "0" to "9". This restricted character set was chosen so that these names can be directly used as in other types of URI including segments of an HTTP path with no special encoding. [RFC5952] contains advice on encoding an IPv6 address in a name.

If either the Base Time or Time value is missing, the missing attribute is considered to have a value of zero. The Base Time and Time values are added together to get the time of measurement. A time of zero indicates that the sensor does not know the absolute time and the measurement was made roughly "now". A negative value is used to indicate seconds in the past from roughly "now". A positive value is used to indicate the number of seconds, excluding leap seconds, since the start of the year 1970 in UTC.

A measurement array MAY be followed by another base object and measurement array. The new base object can add, change, and/or remove base values from the previous base object(s). The new base values are applied to the following measurement arrays. Every base object MUST be followed by a measurement array, and hence base objects are found in the root array at even indexes and measurement arrays at odd indexes.

Representing the statistical characteristics of measurements can be very complex. Future specification may add new attributes to provide better information about the statistical properties of the measurement.

5. Associating Meta-data

SenML is designed to carry the minimum dynamic information about measurements, and for efficiency reasons does not carry more static meta-data about the device, object or sensors. Instead, it is assumed that this meta-data is carried out of band. For web resources using SenML representations, this meta-data can be made available using the CoRE Link Format [RFC6690].

The CoRE Link Format provides a simple way to describe Web Links, and in particular allows a web server to describe resources it is

hosting. The list of links that a web server has available, can be discovered by retrieving the /.well-known/core resource, which returns the list of links in the CoRE Link Format. Each link may contain attributes, for example title, resource type, interface description, and content-type.

The most obvious use of this link format is to describe that a resource is available in a SenML format in the first place. The relevant media type indicator is included in the Content-Type (ct=) attribute.

Further semantics about a resource can be included in the Resource Type and Interface Description attributes. The Resource Type (rt=) attribute is meant to give a semantic meaning to that resource. For example rt="outdoor-temperature" would indicate static semantic meaning in addition to the unit information included in SenML. The Interface Description (if=) attribute is used to describe the REST interface of a resource, and may include e.g. a reference to a WADL description [WADL].

6. JSON Representation (application/senml+json)

Base object variables:

SenML	JSON	Type
Base Name	bn	String
Base Time	bt	Number
Base Units	bu	Number
Version	ver	Number

Measurement or Parameter Entries:

SenML	JSON	Notes
Name	n	String
Units	u	String
Value	v	Floating point
String Value	sv	String
Boolean Value	bv	Boolean
Value Sum	s	Floating point
Time	t	Number
Update Time	ut	Number

All of the data is UTF-8, but since this is for machine to machine communications on constrained systems, only characters with code points between U+0001 and U+007F are allowed which corresponds to the ASCII [RFC0020] subset of UTF-8.

The root content consists of an array with even amount of JSON objects where the first (and then every odd) element is a base object and the second (and then every even) element is a measurements array. The base object MAY contain a "bn" attribute with a value of type string. The object MAY contain a "bt" attribute with a value of type number. The object MAY contain a "bu" attribute with a value of type string. The object MAY contain a "ver" attribute with a value of type number. The object MAY contain other attribute value pairs. The base object MUST be followed by an array. The array MUST have one or more measurement or parameter objects.

If the root array has more than one base object, each following base object modifies the base values using the JSON merge patch format [RFC7396]. That is, base values can be added or modified by defining their new values and existing base values can removed by defining the value as "null".

Inside each measurement or parameter object the "n", "u", and "sv" attributes are of type string, the "t" and "ut" attributes are of type number, the "bv" attribute is of type boolean, and the "v" and "s" attributes are of type floating point. All the attributes are optional, but as specified in Section 4, one of the "v", "sv", or "bv" attributes MUST appear unless the "s" attribute is also present. The "v", and "sv", and "bv" attributes MUST NOT appear together.

Systems receiving measurements MUST be able to process the range of floating point numbers that are representable as an IEEE double-precision floating-point numbers [IEEE.754.1985]. The number of significant digits in any measurement is not relevant, so a reading of 1.1 has exactly the same semantic meaning as 1.10. If the value has an exponent, the "e" MUST be in lower case. The mantissa SHOULD be less than 19 characters long and the exponent SHOULD be less than 5 characters long. This allows time values to have better than micro second precision over the next 100 years.

6.1. Examples

6.1.1. Single Datapoint

The following shows a temperature reading taken approximately "now" by a 1-wire sensor device that was assigned the unique 1-wire address of 10e2073a01080063:

```
[ {}, [{ "n": "urn:dev:ow:10e2073a01080063", "v":23.5 }] ]
```

6.1.2. Multiple Datapoints

The following example shows voltage and current now, i.e., at an unspecified time. The device has an EUI-64 MAC address of 0024befffe804ff1.

```
[{"bn": "urn:dev:mac:0024befffe804ff1/"},
 [ { "n": "voltage", "t": 0, "u": "V", "v": 120.1 },
   { "n": "current", "t": 0, "u": "A", "v": 1.2 } ]
]
```

The next example is similar to the above one, but shows current at Tue Jun 8 18:01:16 UTC 2010 and at each second for the previous 5 seconds.

```
[{"bn": "urn:dev:mac:0024befffe804ff1/",
  "bt": 1276020076,
  "bu": "A",
  "ver": 1},
 [ { "n": "voltage", "u": "V", "v": 120.1 },
   { "n": "current", "t": -5, "v": 1.2 },
   { "n": "current", "t": -4, "v": 1.30 },
   { "n": "current", "t": -3, "v": 0.14e1 },
   { "n": "current", "t": -2, "v": 1.5 },
   { "n": "current", "t": -1, "v": 1.6 },
   { "n": "current", "t": 0,  "v": 1.7 } ]
]
```

Note that in some usage scenarios of SenML the implementations MAY store or transmit SenML in a stream-like fashion, where data is collected over time and continuously added to the object. This mode of operation is optional, but systems or protocols using SenML in this fashion MUST specify that they are doing this. In this situation the SenML stream can be sent and received in a partial fashion, i.e., a measurement entry can be read as soon as it is received and only not when the entire SenML object is complete.

For instance, the following stream of measurements may be sent from the producer of a SenML object to the consumer of that SenML object, and each measurement object may be reported at the time it arrives:

```
[{"bn": "http://[2001:db8::1]",
  "bt": 1320067464,
  "bu": "%RH"},
 [ { "v": 21.2, "t": 0 },
   { "v": 21.3, "t": 10 },
   { "v": 21.4, "t": 20 },
   { "v": 21.4, "t": 30 },
   { "v": 21.5, "t": 40 },
   { "v": 21.5, "t": 50 },
   { "v": 21.5, "t": 60 },
   { "v": 21.6, "t": 70 },
   { "v": 21.7, "t": 80 },
   { "v": 21.5, "t": 90 },
...
```

6.1.3. Multiple Measurements

The following example shows humidity measurements from a mobile device with an IPv6 address 2001:db8::1, starting at Mon Oct 31 13:24:24 UTC 2011. The device also provides position data, which is provided in the same measurement or parameter array as separate entries. Note time is used to for correlating data that belongs together, e.g., a measurement and a parameter associated with it. Finally, the device also reports extra data about its battery status at a separate time.

```
[{"bn": "http://[2001:db8::1]",
  "bt": 1320067464,
  "bu": "%RH"},
 [ { "v": 20.0, "t": 0 },
   { "v": 24.30621, "u": "lon", "t": 0 },
   { "v": 60.07965, "u": "lat", "t": 0 },
   { "v": 20.3, "t": 60 },
   { "v": 24.30622, "u": "lon", "t": 60 },
   { "v": 60.07965, "u": "lat", "t": 60 },
   { "v": 20.7, "t": 120 },
   { "v": 24.30623, "u": "lon", "t": 120 },
   { "v": 60.07966, "u": "lat", "t": 120 },
   { "v": 98.0, "u": "%EL", "t": 150 },
   { "v": 21.2, "t": 180 },
   { "v": 24.30628, "u": "lon", "t": 180 },
   { "v": 60.07967, "u": "lat", "t": 180 } ]
]
```

6.1.4. Collection of Resources

The following example shows how to query one device that can provide multiple measurements. The example assumes that a client has fetched information from a device at 2001:db8::2 by performing a GET operation on http://[2001:db8::2] at Mon Oct 31 16:27:09 UTC 2011, and has gotten two separate values as a result, a temperature and humidity measurement.

```
[{"bn": "http://[2001:db8::2]/",
  "bt": 1320078429,
  "ver": 1},
 [ { "n": "temperature", "v": 27.2, "u": "Cel" },
   { "n": "humidity", "v": 80, "u": "%RH" } ]
]
```

7. CBOR Representation (application/senml+cbor)

The CBOR [RFC7049] representation is equivalent to the JSON representation, with the following changes:

- For compactness, the CBOR representation uses integers for the map keys defined in Table 1. This table is conclusive, i.e., there is no intention to define any additional integer map keys; any extensions will use string map keys.

- For JSON Numbers, the CBOR representation can use integers, floating point numbers, or decimal fractions (CBOR Tag 4); the common limitations of JSON implementations are not relevant for these. For the version number, however, only an unsigned integer is allowed.

Name	JSON label	CBOR label
Version	ver	-1
Base Name	bn	-2
Base Time	bt	-3
Base Units	bu	-4
Name	n	0
Units	u	1
Value	v	2
String Value	sv	3
Boolean Value	bv	4
Value Sum	s	5
Time	t	6
Update Time	ut	7

Table 1: CBOR representation: integers for map keys

8. XML Representation (application/senml+xml)

A SenML object can also be represented in XML format as defined in this section. The following example shows an XML example for the same sensor measurement as in Section 6.1.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<senml xmlns="urn:ietf:params:xml:ns:senml"
       bn="urn:dev:mac:0024befffe804ff1/"
       bt="1276020076"
       ver="1" bu="A">
  <e n="voltage" u="V" v="120.1" />
  <e n="current" t="-5" v="1.2" />
  <e n="current" t="-4" v="1.30" />
  <e n="current" t="-3" v="0.14e1" />
  <e n="current" t="-2" v="1.5" />
  <e n="current" t="-1" v="1.6" />
  <e n="current" t="0" v="1.7" />
</senml>
```

The RelaxNG schema for the XML is:

```
default namespace = "urn:ietf:params:xml:ns:senml"
namespace rng = "http://relaxng.org/ns/structure/1.0"

e = element e {
  attribute n { xsd:string }?,
  attribute u { xsd:string }?,
  attribute v { xsd:float }?,
  attribute sv { xsd:string }?,
  attribute bv { xsd:boolean }?,
  attribute s { xsd:decimal }?,
  attribute t { xsd:int }?,
  attribute ut { xsd:int }?,
  p*
}

senml =
  element senml {
    attribute bn { xsd:string }?,
    attribute bt { xsd:int }?,
    attribute bu { xsd:string }?,
    attribute ver { xsd:int }?,
    e*
  }

start = senml
```

9. EXI Representation (application/senml-exi)

For efficient transmission of SenML over e.g. a constrained network, Efficient XML Interchange (EXI) can be used. This encodes the XML Schema structure of SenML into binary tags and values rather than ASCII text. An EXI representation of SenML SHOULD be made using the strict schema-mode of EXI. This mode however does not allow tag extensions to the schema, and therefore any extensions will be lost in the encoding. For uses where extensions need to be preserved in EXI, the non-strict schema mode of EXI MAY be used.

The EXI header option MUST be included. An EXI schemaID options MUST be set to the value of "a" indicating the scheme provided in this specification. Future revisions to the schema can change this schemaID to allow for backwards compatibility. When the data will be transported over CoAP or HTTP, an EXI Cookie SHOULD NOT be used as it simply makes things larger and is redundant to information provided in the Content-Type header.

The following XSD Schema is generated from the RelaxNG and used for strict schema guided EXI processing.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           targetNamespace="urn:ietf:params:xml:ns:senml"
           xmlns:ns1="urn:ietf:params:xml:ns:senml">

  <xs:element name="e">
    <xs:complexType>
      <xs:attribute name="n" type="xs:string"/>
      <xs:attribute name="u" type="xs:string"/>
      <xs:attribute name="v" type="xs:float"/>
      <xs:attribute name="sv" type="xs:string"/>
      <xs:attribute name="bv" type="xs:boolean"/>
      <xs:attribute name="s" type="xs:decimal"/>
      <xs:attribute name="t" type="xs:int"/>
      <xs:attribute name="ut" type="xs:int"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="senml">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="ns1:e"/>
      </xs:sequence>
      <xs:attribute name="bn" type="xs:string"/>
      <xs:attribute name="bt" type="xs:int"/>
      <xs:attribute name="bu" type="xs:string"/>
      <xs:attribute name="ver" type="xs:int"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following shows a hexdump of the EXI produced from encoding the following XML example. Note that while this example is similar to the first example in Section 6.1.2 in JSON format.

```
<?xml version="1.0" encoding="UTF-8"?>
<senml xmlns="urn:ietf:params:xml:ns:senml"
       bn="urn:dev:ow:10e2073a01080063" >
  <e n="voltage" t="0" v="120.1" u="V" />
  <e n="current" t="0" v="1.2" u="A" />
</senml>
```

Which compresses to the following displayed in hexdump:

```
00000000  a0 30 0d 85 01 d7 57 26  e3 a6 46 57 63 a6 f7 73
00000010  a3 13 06 53 23 03 73 36  13 03 13 03 83 03 03 63
00000020  36 21 2e cd ed 8e 8c 2c  ec a8 00 00 d5 95 88 4c
00000030  02 08 4b 1b ab 93 93 2b  73 a2 00 00 34 14 19 00
00000040  c0
```

The above example used the bit packed form of EXI but it is also possible to use a byte packed form of EXI which can makes it easier for a simple sensor to produce valid EXI without really implementing EXI. Consider the example of a temperature sensor that produces a value in tenths of degrees Celsius over a range of 0.0 to 55.0. It would produce an XML SenML file such as:

```
<?xml version="1.0" encoding="UTF-8"?>
<senml xmlns="urn:ietf:params:xml:ns:senml"
       bn="urn:dev:ow:10e2073a01080063" >
  <e n="temp" v="23.1" u="Cel" />
</senml>
```

The compressed form, using the byte alignment option of EXI, for the above XML is the following:

```
00000000  a054c9006c200000 1d75726e3a646576 |.T..l ...urn:dev|
00000010  3a6f773a31306532 3037336130313038 |:ow:10e2073a0108|
00000020  3030363304010674 656d70030543656c |0063...temp..Cel|
00000030  0100e70101000102                  |........|
```

A small temperature sensor devices that only generates this one EXI file does not really need an full EXI implementation. It can simple hard code the output replacing the one wire device ID starting at byte 0x14 and going to byte 0x23 with it's device ID, and replacing the value "0xe7 0x01" at location 0x32 to 0x33 with the current temperature. The EXI Specification [W3C.REC-exi-20110310] contains the full information on how floating point numbers are represented, but for the purpose of this sensor, the temperature can be converted to an integer in tenths of degrees (231 in this example). EXI stores 7 bits of the integer in each byte with the top bit set to one if there are further bytes. So the first bytes at location 0x32 is set to low 7 bits of the integer temperature in tenths of degrees plus 0x80. In this example 231 & 0x7F + 0x80 = 0xE7. The second byte at location 0x33 is set to the integer temperature in tenths of degrees right shifted 7 bits. In this example 231 >> 7 = 0x01.

10. Usage Considerations

The measurements support sending both the current value of a sensor as well as the an integrated sum. For many types of measurements, the sum is more useful than the current value. For example, an

electrical meter that measures the energy a given computer uses will typically want to measure the cumulative amount of energy used. This is less prone to error than reporting the power each second and trying to have something on the server side sum together all the power measurements. If the network between the sensor and the meter goes down over some period of time, when it comes back up, the cumulative sum helps reflect what happened while the network was down. A meter like this would typically report a measurement with the units set to watts, but it would put the sum of energy used in the "s" attribute of the measurement. It might optionally include the current power in the "v" attribute.

While the benefit of using the integrated sum is fairly clear for measurements like power and energy, it is less obvious for something like temperature. Reporting the sum of the temperature makes it easy to compute averages even when the individual temperature values are not reported frequently enough to compute accurate averages. Implementors are encouraged to report the cumulative sum as well as the raw value of a given sensor.

Applications that use the cumulative sum values need to understand they are very loosely defined by this specification, and depending on the particular sensor implementation may behave in unexpected ways. Applications should be able to deal with the following issues:

1. Many sensors will allow the cumulative sums to "wrap" back to zero after the value gets sufficiently large.

2. Some sensors will reset the cumulative sum back to zero when the device is reset, loses power, or is replaced with a different sensor.

3. Applications cannot make assumptions about when the device started accumulating values into the sum.

Typically applications can make some assumptions about specific sensors that will allow them to deal with these problems. A common assumption is that for sensors whose measurement values are always positive, the sum should never get smaller; so if the sum does get smaller, the application will know that one of the situations listed above has happened.

11. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "RFC-AAAA" with the RFC number of this specification.

11.1. Units Registry

IANA will create a registry of unit symbols. The primary purpose of this registry is to make sure that symbols uniquely map to give type of measurement. Definitions for many of these units can be found in [NIST811] and [BIPM].

All the registry entries in Table 2 use "v" (numeric) values. New entries allocated in the registry must define what kind of values they use.

Symbol	Description	Reference
m	meter	RFC-AAAA
kg	kilogram	RFC-AAAA
s	second	RFC-AAAA
A	ampere	RFC-AAAA
K	kelvin	RFC-AAAA
cd	candela	RFC-AAAA
mol	mole	RFC-AAAA
Hz	hertz	RFC-AAAA
rad	radian	RFC-AAAA
sr	steradian	RFC-AAAA
N	newton	RFC-AAAA
Pa	pascal	RFC-AAAA
J	joule	RFC-AAAA
W	watt	RFC-AAAA
C	coulomb	RFC-AAAA
V	volt	RFC-AAAA
F	farad	RFC-AAAA
Ohm	ohm	RFC-AAAA
S	siemens	RFC-AAAA
Wb	weber	RFC-AAAA
T	tesla	RFC-AAAA
H	henry	RFC-AAAA
Cel	degrees Celsius	RFC-AAAA
lm	lumen	RFC-AAAA
lx	lux	RFC-AAAA
Bq	becquerel	RFC-AAAA
Gy	gray	RFC-AAAA
Sv	sievert	RFC-AAAA
kat	katal	RFC-AAAA
pH	pH acidity	RFC-AAAA
%	Value of a switch (note 1)	RFC-AAAA
count	counter value	RFC-AAAA
%RH	Relative Humidity	RFC-AAAA
m2	area	RFC-AAAA

l	volume in liters	RFC-AAAA
m/s	velocity	RFC-AAAA
m/s2	acceleration	RFC-AAAA
l/s	flow rate in liters per second	RFC-AAAA
W/m2	irradiance	RFC-AAAA
cd/m2	luminance	RFC-AAAA
Bspl	bel sound pressure level	RFC-AAAA
bit/s	bits per second	RFC-AAAA
lat	degrees latitude (note 2)	RFC-AAAA
lon	degrees longitude (note 2)	RFC-AAAA
%EL	remaining battery energy level in percents	RFC-AAAA
EL	remaining battery energy level in seconds	RFC-AAAA
beat/m	Heart rate in beats per minute	RFC-AAAA
beats	Cumulative number of heart beats	RFC-AAAA

Table 2

- Note 1: A value of 0.0 indicates the switch is off while 100.0 indicates on.

- Note 2: Assumed to be in WGS84 unless another reference frame is known for the sensor.

New entries can be added to the registration by either Expert Review or IESG Approval as defined in [RFC5226]. Experts should exercise their own good judgment but need to consider the following guidelines:

1. There needs to be a real and compelling use for any new unit to be added.

2. Units should define the semantic information and be chosen carefully. Implementors need to remember that the same word may be used in different real-life contexts. For example, degrees when measuring latitude have no semantic relation to degrees when measuring temperature; thus two different units are needed.

3. These measurements are produced by computers for consumption by computers. The principle is that conversion has to be easily be done when both reading and writing the media type. The value of a single canonical representation outweighs the convenience of easy human representations or loss of precision in a conversion.

4. Use of SI prefixes such as "k" before the unit is not allowed. Instead one can represent the value using scientific notation such a 1.2e3.

5. For a given type of measurement, there will only be one unit type defined. So for length, meters are defined and other lengths such as mile, foot, light year are not allowed. For most cases, the SI unit is preferred.

6. Symbol names that could be easily confused with existing common units or units combined with prefixes should be avoided. For example, selecting a unit name of "mph" to indicate something that had nothing to do with velocity would be a bad choice, as "mph" is commonly used to mean miles per hour.

7. The following should not be used because the are common SI prefixes: Y, Z, E, P, T, G, M, k, h, da, d, c, n, u, p, f, a, z, y, Ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi.

8. The following units should not be used as they are commonly used to represent other measurements Ky, Gal, dyn, etg, P, St, Mx, G, Oe, Gb, sb, Lmb, ph, Ci, R, RAD, REM, gal, bbl, qt, degF, Cal, BTU, HP, pH, B/s, psi, Torr, atm, at, bar, kWh.

9. The unit names are case sensitive and the correct case needs to be used, but symbols that differ only in case should not be allocated.

10. A number after a unit typically indicates the previous unit raised to that power, and the / indicates that the units that follow are the reciprocal. A unit should have only one / in the name.

11. A good list of common units can be found in the Unified Code for Units of Measure [UCUM].

11.2. Media Type Registration

The following registrations are done following the procedure specified in [RFC6838] and [RFC7303].

Note to RFC Editor: Please replace all occurrences of "RFC-AAAA" with the RFC number of this specification.

11.2.1. senml+json Media Type Registration

Type name: application

Subtype name: senml+json

Required parameters: none

Optional parameters: none

Encoding considerations: Must be encoded as using a subset of the encoding allowed in [RFC7159]. Specifically, only the ASCII [RFC0020] subset of the UTF-8 characters are allowed. This simplifies implementation of very simple system and does not impose any significant limitations as all this data is meant for machine to machine communications and is not meant to be human readable.

Security considerations: Sensor data can contain a wide range of information ranging from information that is very public, such the outside temperature in a given city, to very private information that requires integrity and confidentiality protection, such as patient health information. This format does not provide any security and instead relies on the transport protocol that carries it to provide security. Given applications need to look at the overall context of how this media type will be used to decide if the security is adequate.

Interoperability considerations: Applications should ignore any JSON key value pairs that they do not understand. This allows backwards compatibility extensions to this specification. The "ver" field can be used to ensure the receiver supports a minimal level of functionality needed by the creator of the JSON object.

Published specification: RFC-AAAA

Applications that use this media type: The type is used by systems that report electrical power usage and environmental information such as temperature and humidity. It can be used for a wide range of sensor reporting systems.

Additional information:

Magic number(s): none

File extension(s): senml

Macintosh file type code(s): none

Person & email address to contact for further information: Cullen Jennings <fluffy@iii.ca>

Intended usage: COMMON

Restrictions on usage: None

Author: Cullen Jennings <fluffy@iii.ca>

Change controller: IESG

11.2.2. senml+cbor Media Type Registration

Type name: application

Subtype name: senml+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: TBD

Security considerations: TBD

Interoperability considerations: TBD

Published specification: RFC-AAAA

Applications that use this media type: The type is used by systems that report electrical power usage and environmental information such as temperature and humidity. It can be used for a wide range of sensor reporting systems.

Additional information:

Magic number(s): none

File extension(s): senml

Macintosh file type code(s): none

Person & email address to contact for further information: Cullen Jennings <fluffy@iii.ca>

Intended usage: COMMON

Restrictions on usage: None

Author: Cullen Jennings <fluffy@iii.ca>

Change controller: IESG

11.2.3. senml+xml Media Type Registration

Type name: application

Subtype name: senml+xml

Required parameters: none

Optional parameters: none

Encoding considerations: TBD

Security considerations: TBD

Interoperability considerations: TBD

Published specification: RFC-AAAA

Applications that use this media type: TBD

Additional information:

Magic number(s): none

File extension(s): senml

Macintosh file type code(s): none

Person & email address to contact for further information: Cullen Jennings <fluffy@iii.ca>

Intended usage: COMMON

Restrictions on usage: None

Author: Cullen Jennings <fluffy@iii.ca>

Change controller: IESG

11.2.4. senml-exi Media Type Registration

Type name: application

Subtype name: senml-exi

Required parameters: none

Optional parameters: none

Encoding considerations: TBD

Security considerations: TBD

Interoperability considerations: TBD

Published specification: RFC-AAAA

Applications that use this media type: TBD

Additional information:

Magic number(s): none

File extension(s): senml

Macintosh file type code(s): none

Person & email address to contact for further information: Cullen Jennings <fluffy@iii.ca>

Intended usage: COMMON

Restrictions on usage: None

Author: Cullen Jennings <fluffy@iii.ca>

Change controller: IESG

11.3. XML Namespace Registration

This document registers the following XML namespaces in the IETF XML registry defined in [RFC3688].

URI: urn:ietf:params:xml:ns:senml

Registrant Contact: The IESG.

XML: N/A, the requested URIs are XML namespaces

11.4. CoAP Content-Format Registration

IANA is requested to assign CoAP Content-Format IDs for the SenML media types in the "CoAP Content-Formats" sub-registry, within the "CoRE Parameters" registry [RFC7252]. All IDs are assigned from the "Expert Review" (0-255) range. The assigned IDs are show in Table 3.

Media type	ID
application/senml+json	TBD
application/senml+cbor	TBD
application/senml+xml	TBD
application/senml-exi	TBD

Table 3: CoAP Content-Format IDs

12. Security Considerations

See Section 13. Further discussion of security properties can be found in Section 11.2.

13. Privacy Considerations

Sensor data can range from information with almost no security considerations, such as the current temperature in a given city, to highly sensitive medical or location data. This specification provides no security protection for the data but is meant to be used inside another container or transport protocol such as S/MIME or HTTP with TLS that can provide integrity, confidentiality, and authentication information about the source of the data.

14. Acknowledgement

We would like to thank Lisa Dusseault, Joe Hildebrand, Lyndsay Campbell, Martin Thomson, John Klensin, Bjoern Hoehrmann, Carsten Bormann, and Christian Amsuess for their review comments.

The CBOR Representation text was contributed by Carsten Bormann.

15. References

15.1. Normative References

[BIPM] Bureau International des Poids et Mesures, "The International System of Units (SI)", 8th edition, 2006.

[IEEE.754.1985] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

[NIST811] Thompson, A. and B. Taylor, "Guide for the Use of the International System of Units (SI)", NIST Special Publication 811, 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7303] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, DOI 10.17487/RFC7303, July 2014, <http://www.rfc-editor.org/info/rfc7303>.

[RFC7396] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396, DOI 10.17487/RFC7396, October 2014, <http://www.rfc-editor.org/info/rfc7396>.

[W3C.REC-exi-20110310]
Schneider, J. and T. Kamiya, "Efficient XML Interchange (EXI) Format 1.0", World Wide Web Consortium Recommendation REC-exi-20110310, March 2011, <http://www.w3.org/TR/2011/REC-exi-20110310>.

15.2. Informative References

[I-D.arkko-core-dev-urn]
Arkko, J., Jennings, C., and Z. Shelby, "Uniform Resource Names for Device Identifiers", draft-arkko-core-dev-urn-03 (work in progress), July 2012.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <http://www.rfc-editor.org/info/rfc20>.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, DOI 10.17487/RFC2141, May 1997, <http://www.rfc-editor.org/info/rfc2141>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <http://www.rfc-editor.org/info/rfc5952>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <http://www.rfc-editor.org/info/rfc6690>.

[UCUM] Schadow, G. and C. McDonald, "The Unified Code for Units of Measure (UCUM)", Regenstrief Institute and Indiana University School of Informatics, 2013, <http://unitsofmeasure.org/ucum.html>.

[WADL] Hadley, M., "Web Application Description Language (WADL)", 2009, <http://java.net/projects/wadl/sources/svn/content/trunk/www/wadl20090202.pdf>.

Authors' Addresses

Cullen Jennings
Cisco
400 3rd Avenue SW
Calgary, AB T2P 4H2
Canada

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Zach Shelby
ARM
150 Rose Orchard
San Jose 95134
USA

Phone: +1-408-203-9434
Email: zach.shelby@arm.com

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Ari Keranen
Ericsson
Jorvas 02420
Finland

Email: ari.keranen@ericsson.com