ENGINEERING WHITE PAPER

CoAP Information Model for Enterprise IoT

Author(s):
John Parello, Senior Technical Leader CSG (jparello@cisco.com)
Padmanabhan Ramanujam,Technical Leader, CSG (pramanuj@cisco.com)

Contributors:
TBD

Revisions:

0.01	Summary for Submission	02/22/2015

1 Abstract

This document gives an overview of an Information Model we propose to be used with a **CoAP Proxy Server** (**CPS**). The CPS will be implemented as a standard CoAP server using UDP transport. In an effort to get a consisted view of Internet of Things (IoT) endpoints as they connect to a proxy, we propose an information model and a corresponding JSON data model based on the Sensor Markup Languauge (SENML). SENM is a draft proposed by the CORE IETF group.

We propose here extending the SENML to be a general result set information model for endpoints. We do this by generalizing the SENML for sensors and actuators with a new classification of measurements. We also propose connectivity information linked to the endpoint through a unique uuid for every endpoint.

Keywords:
CoAP, IoT, IoE, EIoT, Lighting, LED, PoE, Enterprise Management, BMS,

Submission
This paper is submitted to the **IoT Semantic Interoperability Workshop 2016**

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] when they appear in ALL CAPS. These words may also appear in this document in lowercase, absent their normative meanings

2 Background: A CoAP-to-CoAP Proxy Server

We are expecting to have CoAP proxy servers that run a standard CoAP server. The CoAP server will additionally proxy for registered endpoints as well as acting as a server for resources for local resources.

The CPS will listen on the resource discover port (default 5683 and 5684 for DTLS) and register endpoints advertising their resource in the CoRE Link Format [rfc6690].

The CPS will store resource links and make them available via the standard **`/.well-known/core`** URI. Endpoints will be modeled as resources so that the CPS can act as a registry for endpoints and their resources.

CPS will be able to register themselves in the resource directory of other CPS'. A CPS registering with another CPS will only register itself (not the other r resources that have register to it). This will enable deployments to create a Zone of CoAP servers. An example of such a setup can be seen as:

2.1 Discovery / Registry

Typically a CoAP resource directory would index resource types of the device hosting the CoAP server. The resource directory of a CPS will be organized by creating a node entry for each endpoint or CPS registered[sensc9]. Each node would then contain a list of resource types the node implements.

The CPS resulting resource directory would be a tree-structure where the endpoint is at the top of the resource and resource descriptions are the branches.

The root of the tree will be the **`.well-known/core`** of the CPS. This can be seen as follow:

3 Information Models

We expect endpoints to track an information (model) and present that information via resources (view). This section will describe the information model in a generic way that does not dictate storage or implementation for the endpoint - just what the endpoint should track. Following the model we describe a resource view of the information. Resources will present the information using a structure based on SenML[senml]

The model describes the minimal set of information needed for and endpoint (device) to be usefully accessed from the **CPS**. The model describes information pertaining to the **identity**, **inventory**, **context**, **network**, and **measurements** for the endpoint device. Measurements are modeled as **sensor** (readable) and **actuator** (writable) information.

For the modeling we assume:

- An endpoint implements a CoAP server.
- The endpoint will contain inventory and network information.
- The endpoint may contain multiple sensors/actuator that are components of the endpoint.
- The components contained on the endpoint (including the endpoint itself) will contain identity, context, sensor (measurement) and actuator (settings).
- The endpoint and components will contain a unique identifier (uuid)

.

3.1 Endpoint Model

The following syntax based on Unified Modeling Language (UML) [ISO-IEC-19501-2005] will be used to represent the information model..

```
UML Construct           Equivalent Notation
--------------------    ----------------------------------
```

Notes	`// Notes`
Class (Generalization)	`CLASS name {member..}`
Subclass (Specialization)	`CLASS subclass  EXTENDS superclass {member..}`
Class Member (Attribute)	`attribute : type`

This model does not dictate an implementation (that could be a sparser embodiment of the model). It is a reference of the information that and endpoint would track and delver via resources.

```
CLASS Endpoint {
  device              : Device;
  numberOfComponents  : integer;
  components          : Component[ ];
}

CLASS Component {
  identity    : Identity;
  context     : Context;
  location    : Location;
  measurement : Measurement;
  permission  : string;
}


CLASS Device {
  identity  : Identity;
  context   : Context;
  location  : Location;
  inventory : Inventory;
  network   : Network;
}


CLASS Context  {
  domainName      : string;
  roleDescription : string;
  keywords        : string[ ];
  importance      : integer;
}

CLASS Location {
  specifier : string;
  size      : number;
  payload   : byte[ ];

}


CLASS Inventory {
  hardwareRevision : string;
  firmwareRevision : string ;
  softwareRevision : string ;
  serialNumber     : string ;
  manufacturer     : string;
  model            : string;
}


CLASS Identity {
 entPhysicalUUID  : string; //uuid
 entPhysicalNAme  : string;
 entPhysicalClass : string;
 alternateKey     : string;
}


CLASS Network {
  mgmtMacAddress    : string;
  mgmtAddressType   : InetAddressType;
  mgmtAddress       : string;
  mgmtDNSName       : string ;
  lldp              : Lldp ;
}
```

```
CLASS Network {
  mgmtMacAddress    : string;
  mgmtAddressType   : InetAddressType;
  mgmtAddress       : string;
  mgmtDNSName       : string ;
  lldp              : Lldp ;
}
```

```
CLASS Lldp  {
  lldpRemLocalPortNum   : integer;
  lldpRemIndex          : integer;
  lldpRemManAddrSubtype : InetAddressType;
  lldpRemManAddr        : string;
}

CLASS Measurement {
  mclass      : string;
  value       : number;
  units       : string;
  multiplier  : integer;
  accuracy    : integer;
  caliber     : integer;
  sensorTime  : Time;
  control     : ControlEntry;
}

CLASS ControlEntry {
  percentPower               : integer;
  percentIntensity           : integer;
  incrementalPercentIntensity: integer;
  adjustTime                 : integer;
  adjustChangeRate           : integer;
}

CLASS Sensors
   //See SenML
}

CLASS Actuators {
  // see SenML + ControlEntry
}
```

3.2 Sensor Markup Language

Based on draft-jennings-core-senml-01

SenML is a CoAP friendly format expressed in JSON/CBOR that can be used to encapsulate a request from the view (resources). The basic format of SenML is a set of time and version information followed by a collection (array) of measurements.

The information based on SenML will be formatted as JSON/CBOR representation. The XML/XMI formats described in the draft will not be use.

We describe a scheme that uses the basic SenML format to encapsulate result requests for **identity**, **inventory**, **context**, **network** as well as **sensor** / **actuator** resources.

The idea being that there would be request attributes followed by a collection of information for each component.

The benefit of using this type of structure is that the SenML is used as a result set container for information. The collection in SenML would be scoped for components (on a device; attached to a switch; the entire domain) and bound by type to the resource request (identity, inventory, sensor etc).

For updates we propose adding a control entry to SenML that can be presented when POST/PUT/PATCH-ing a resource.

The following describes the SenML per the IETF drafts with three modification proposals:

- The SenML measurement attributes lack fields that we've taken from the IETF EMAN Measurements. These should be added to the schema for a measurement and can optionally be included,
- A control entry used for updates
- A modified entry collection allowing multiple types of objects (bound when a specific resource is requested)

Additional Attributes for Measurements

SenML	JSON	Type	Description
CLASS	cl	string	The type of sensor(See listing)
MULTIPLIER	m	number	A sensor multiplier representing the SI exponents in the range -24..24 of the Value (v)
PERMISSION	p	string	Two characters representing the r (read permission), w (write permission) respectively with a dash representing not allowed
ACCURACY	a	number	A value > 0 representing the hundreds of a percent of accuracy of the reading
CALIBER	ca	number	0 = actual, 1 = static, 2 = estimated Values > 2 can further indicate the estimation method

Class Registry

CLASS	UNITS	DESCRIPTION
color	rgbw	rgbw - Quadruple of integers color mixture
color	K	kelvin -The black body correlated color temperature
power	W	watts
energy	Wh	watts-hours
distance	m	meters
weight	g	grams
time	s	seconds
area	m2	meters squares
velocity	m/s	meters per second
acceleration	m/s2	meters per second squared
humidity	%RH	relative humidity
temperature	Cel	Celsius
temperature	K	Kelvin
count		Integer
light	lx	lux
light	lm	lumen
light	cd	candela
boolean		Boolean
pressure	Pa	Pascal

Control Entry

There is no specification for the control of sensor information in SenML. From our work on PoC's and ecosystem trials we've found that a control element entry would be needed. The control element would contain attributes to describe a change. These attributes would need to be added to SenML. The scheme would be to use the SenML root variable and add a control entry (ce) with the following format:

Name	JSON	Type	Description
PERCENT_POWER	pp	number	Hundredths of Percent of maximum power
PERCENT_INTENSITY	pi	number	Hundredths of Percent of maximum Absolute intensity (Where intensity is the functional output of the EndPoint)
INCREMENTAL_PERCENT_INTENSITY	ii	number	Signed hundredths of a percent to change from the present intensity
ADJUST_TIME	at	number	Time period (milliseconds) over which to adjust a setting.
ADJUST_CHANGE_RATE	ar	number	Hundredths of percent to change per minute

Control Entry:

SenML	JSON	Notes	
Percent Power	pp	String	*
% Intensity	pi	Number	*
Inc% Intensity	ii	Number	*
Adjust Time	at	Number	*
Adjust Rate	ar	Number	*
Control Name	cn	String	*

Object Type for (e) array

Root Variables:

SenML	JSON	Type	
Base Name	bn	String	
Base Time	bt	Number	
Base Units	bu	Number	
Version	ver	Number	
Measurement or Object	e	Array	*
Control Entry	ce	Object	*

Measurement or Parameter Entries:

SenML	JSON	Notes	
Name	n	String	
Units	u	String	
Integer Value	vi	Number	*
Float Value	v,vf	Floating point	~
String Value	sv,vs	String	~
Boolean Value	bv,vb	Boolean	~
Value Sum	s	Floating point	
Time	t	Number	
Class	cl	String	*
Multiplier	m	Number	*
Permission	p	String	*
Accuracy	a	enum	*
Caliber	ca	enum	*

3.3 Resource View

The resource view will map a resource identifier to a model class. A collection of objects for resource will be return enveloped in a SenML object. The (e)ntry collection will be bound to the class specified by the resource. Each resource will be listed here mapped to the model class with an example output. The JSON/CBOR data models should use the short names for attributes. The short names for the classes we've added to the SenML container are all 4 characters in length listed as a note.

3.3.1 /cisco/identity

Based on RFC7326/RFC7461 Identity information. All values are ReadOnly

```
CLASS Identity {

               uuid     : string  // uuid
      entPhysicalName   : string  // enam
      entPhysicalClass  : string  // ecla
      alternateKey      : string  // akey

}
```

3.3.2 /cisco/context

Based on RFC7326/RFC7461 Identity information. All values are ReadWrite

```
CLASS Context {
      uuid             : string  // uuid
      domainName       : string  // domn
      roleDescription  : string  // role
      keywords         : array   // keyw
      importance       : integer // impo
}
```

3.3.3 /cisco/location

Since no one standard for civic, placement, elevation, angle or geo location can generalize the placement of a sensor, the /cisco/location will encapsulate any scheme the endpoint could provide. The value is ReadOnly

```
CLASS Location {
      uuid      : string  // uuid
      specifier : string  // spec
      size      : integer // size
      payload   : byte[]  // payl
}
```

3.3.4 /cisco/inventory

```
CLASS Inventory {
  uuid             : string // uuid
  hardwareRevision : string // hwrv
  firmwareRevision : string // fwrv
  softwareRevision : string // swrv
  serialNumber     : string // snum
  manufacturer     : string // manu
  model            : string // modl
}
```

3.3.5 /cisco/network

```
CLASS Network { {
  uuid               : string // uuid
  mgmtMacAddress     : string // mmac
  mgmtAddressType    : string // madt
  mgmtAddress        : string // madr
  mgmtDNSName        : string // mdns
}
```

3.3.6 /cisco/sensor

Based on draft-jennings-core-senml-01.

The `/sensor` resource is considered ReadOnly with only a `GET` operation permitted.
The senml object can contain one or more measurements.
The `/sensor` resource will return an entry `(e) array` of sensor measurements for each component on the endpoint per the SenML spec and modifications proposed here..

3.3.7 /cisco/actuator

Based on on draft-jennings-core-senml-01.

The `/actuator` resource is considered ReadWrite with GET, PUT, and PATCH operations allowed. A `POST` or `DELETE` would only apply for observable or other transient or physical attributes modeled on the endpoint. Observable or transient models are not defined here and are left to endpoints to describe as needed.

The `/actuator` information would be modeled with the same construct as `/sensor`. A `GET` operation for actuator would return the same measurement array (e) as `/sensor` with the exception that the array would only contain elements that are writeable.

A control entry (ce) would be added to a root variable that would be applied when modifying the endpoint with information from the element array (e).

3.3.8 CBOR Label (sensor/actuator)

```
The labels specified in draft-jennings-core-senml-01 will be used

+---------------------------+------------+------------+
|                      Name | JSON label | CBOR label |
+---------------------------+------------+------------+
```

Version	ver	-1	
Measurement or Parameters	e	-2	
Base Name	bn	-3	
Base Time	bt	-4	
Base Units	bu	-5	
Name	n	0	
Units	u	1	
(Float)Value	v,vf	2	
String Value	sv,vs	3	
Boolean Value	bv,vb	4	
Value Sum	s	5	
Time	t	6	
Update Time	ut	7	
Intger Value	vi		*
uuid	id		*
Control Entry	ce		*
Class	cl		*
Multiplier	m		*
Permission	p		*
Accuracy	a		*
Caliber	ca		*
Percent Power	pp		*
% Intensity	pi		*
Inc% Intensity	ii		*
Adjust Time	at		*
Adjust Rate	ar		*
Control Name	cn		*

4 Resource Examples

4.1 JSON and CBOR

CBOR is a binary encoding of JSON. This document shows all information using JSON notation because it's readable and text. The equivalency of JSON versus CBOR is specified in RFC7049. A useful site for testing and checking CBOR information can be found at `http://cbor.me`

The following shows an example of the `/cisco/context` attributes encoded in both JSON and CBOR (without labels) just to illustrate some encoding.

JSON

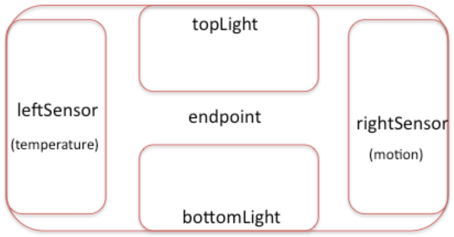
```
{   "Context" : {    "domn"     : "MyDomain"
                   , "role"     : "Decorative Lighting"
                   , "keyw"     : [ "pretty", "non-critical"]
                   , "impo"     : 25
                }
}
```

CBOR

```
a1                                      # map(1)
   67                                   # text(7)
      436f6e74657874                    # "Context"
   a4                                   # map(4)
      64                                # text(4)
         646f6d6e                       # "domn"
      68                                # text(8)
         4d79446f6d61696e               # "MyDomain"
      64                                # text(4)
         726f6c65                       # "role"
      73                                # text(19)
         4465636f726174697665204c69676874696e67 # "Decorative Lighting"
      64                                # text(4)
         6b657977                       # "keyw"
      82                                # array(2)
         66                             # text(6)
            707265747479                # "pretty"
         6c                             # text(12)
            6e6f6e2d637269746963616c    # "non-critical"
      64                                # text(4)
         696d706f                       # "impo"
      18 19                             # unsigned(25)
```

4.2 Reference Endpoint

For illustration the following is a reference endpoints: A lighting fixture with two controllable lights and two sensors for motion and temperature.



The payload will be described using JSON but would minimally be presented in CBOR in transmission using CBOR Labels where defined.

/cisco/identity

```
{
  "e":[
       {"uuid":"1700","enam":"light17","ecla":3}
     , {"uuid":"1717","enam":"toplight","ecla":9}
     , {"uuid":"1718","enam":"bottomlight","ecla":9}
     , {"uuid":"1719","enam":"leftSensor","ecla":8}
     , {"uuid":"1720","enam":"rightSensor","ecla":8}
   ]
  , "bn" : "urn:example"
  , "bt" : 1276020076
  , "ver": 1
}
```

/cisco/inventory

```
{
  "e":[
       {“uuid”:”1700”,"hwrv":"1.1.1","fwrv":"2.2.2"
                      ,"swrv" :"3.3.3","snum" : "1717"
                      ,"manu":"example","modl":"AcmeBrightLight",
]
  , "bn" : "urn:example"
  , "bt" : 1276020076
  , "ver": 1
}
```

/cisco/network

```
{
  "e":[
       {“uuid”:”1700”,"mmac":"0a:0b:0c:0d:0e:0f","madr":"10.10.10.10",
                       "mdns":"light17.exmaple.com"
       }
]
  , "bn" : "urn:example"
  , "bt" : 1276020076
  , "ver": 1
}
```

/cisco/context

```
{
  "e":[
       {“uuid”:”1700”,“domn”:”LondonEye”,”role”:”Decorative Item”,”keyw”:”pretty”}
     , {“uuid”:”1717”,”role”:”Decorative Light”,keyw”:”pretty”,”impo”:20}
     , {“uuid”:”1718”,”role”:”Spot Light”,”keyw”:”task”,”impo”:20}
     , {“uuid”:”1719”,”role”:”Temperature Sensor”,”keyw”:”osha”,”impo”:90}
     , {“uuid”:”1720”,”role”:”Occupancy Sensor”,”keyw”:”lighting”,”impo”:90}
]
  , "bn" : "urn:example"
  , "bt" : 1276020076
  , "ver": 1
}
```

/cisco/location

```
{
  "e":[
       {“uuid”:”1700”, “spec”:”goog”, “payl”:”x:344:y:400:z400”
     , {“uuid”:”1717”, “spec”:”goog”, “payl”:”x:400:y:400:z400”
]
  , "bn" : "urn:example"
  , "bt" : 1276020076
  , "ver": 1
}
```

/cisco/sensor

```
{
  "e":[
       {“uuid”:”1717”,"n":"toplight", "cl":"color", "sv":"00FF0000", "u":"rgbw", "p":"rw"}
     , {“uuid”:”1718”,"n":"bottomlight", "cl":"color", "sv":"FF000000", "u":"rgbw", "p":"rw"}
     , {“uuid”:”1719”,"n":"leftSensor", "cl":"temperature", "v":20, "u":"C", "p":"r"}
     , {“uuid”:”1720”,"n":"rightSensor", "cl":"count", "v":17, "p":"r", "t":1276020080}
   ]
  , "bn" : "urn:example"
  , "bt" : 1276020076
  , "ver": 1
}
```

/cisco/actuator

```
{
  "e":[
       {“uuid”:”1717”,"n":"toplight", "cl":"color", "sv":"FF0000000", "u":"rgbw", "p":"rw"}
       {“uuid”:”1718”,"n":"bottomlight", "cl":"color", "sv":"FFFF0000", "u":"rgbw", "p":"rw"}
  ]
  ,“ce”  : [{ uuid:”1717” “cn” : “scene17”, “pi” : 90, “at” : 3000 }
            { uuid:”1718” “cn” : “scene18”, “pi” : 90, “at” : 2000 }
           ]
  , "bn" : "urn:example"
  , "bt" : 1276020081
  , "ver": 1
}


{
  "e":[
       {“uuid”:”1717”,"n":"toplight", "cl":"color", "sv":"FF0000000", "u":"rgbw", "p":"rw"}
       {“uuid”:”1718”,"n":"bottomlight", "cl":"color", "sv":"FF000000", "u":"rgbw", "p":"rw"}
  ]
  ,“ce”  : [{ “cn” : “scene17”, “pi” : 90, “at” : 3000 }]

  , "bn" : "urn:example"
  , "bt" : 1276020081
  , "ver": 1
}
```

5 References

NORMATIVE

[coap] Shelby, Z.; Hartke, K.; Bormann, C. The Constrained Application Protocol (CoAP); RFC 7252; IETF, October 2014

[cew] J Parello, R Saville, S Kramling, “Cisco Validated Designs”, September 2010, http://www.cisco.com/en/US/docs/solutions/Enterprise/Borderless_Networks/Energy_Management/energywisedg.html

[eman] IETF Energy Management (EMAN) Charter, Sept 2010, https://datatracker.ietf.org/wg/eman/charter/

[eman-model] Energy Management Framework, Dec 2010, RFC7326

[eman-mib] Energy Management: MIB, RFC7461

[sensc9] Ludovici, Calversa, “A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications”, 2015, ISSN 1424-8220

[ianacbor] https://www.iana.org/assignments/media-types/application/cbor [group] https://tools.ietf.org/html/draft-ietf-core-groupcomm-25

White Paper on Cisco UPOE : http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-4500-series-switches/white_paper_c11-670993.pdf

Cisco SNMP reference Link to LLDP MIB : http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2&mibName=LLDP-MIB

INFORMATIVE

[uk-tsb] United Kingdom Technology Strategies Board, UK, 2014, https://www.innovateuk.org, LaaS

[ieee] B Claise, J Parello, "EMAN: Energy-Management Activities at the IETF", Internet Computing IEEE Volume 17 Issue 3, May 2013, http://dx.doi.org/10.1109/MIC.2013.49

RELEVANT STANDARDS or DRAFTS

Regarding Singletons
RFC7353 - The Constrained Application Protocol (CoAP)
RFC6690 - Constrained RESTful Environments (CoRE) Link Format

Regarding Group Communication
RFC7390 (Experimental) – Group Communication for CoAP
draft-ietf-core-resource-directory-02
draft-ietf-core-groupcomm-25

Regarding Information Models
RFC6933 – UUID for Entities
RFC7326 – Information Model for Energy Management (context)
RFC7461 – Data Model for Energy Object Context
[senml] draft-jennings-core-senml-01

Regarding Encoding
RFC7049 - Concise Binary Object Representation (CBOR)