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
}
```