CoRE
Internet-Draft
Intended status: Standards Track
Expires: August 3, 2012

Z. Shelby
Sensinode

January 31, 2012

# CoRE Interfaces
# draft-shelby-core-interfaces-01

## Abstract

This document defines well-known REST interface descriptions for Batch, Sensor, Parameter and Actuator types for use in contrained web servers using the CoRE Link Format. A short reference is provided for each type that can be efficiently included in the interface description attribute of the CoRE Link Format. These descriptions are intended to be for general use in resource designs or for inclusion in more specific interface profiles.

## 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 August 3, 2012.

Table of Contents

## 1. Introduction

The Constrained RESTful Environments (CoRE) working group aims at realizing the REST architecture in a suitable form for the most constrained nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN). CoRE is aimed at machine-to-machine (M2M) applications such as smart energy and building automation.

The discovery of resources offered by a constrained server is very important in machine-to-machine applications where there are no humans in the loop and static interfaces result in fragility. The discovery of resources provided by an HTTP Web Server is typically called Web Linking [RFC5988]. The use of Web Linking for the description and discovery of resources hosted by constrained web servers is specified by the CoRE Link Format [I-D.ietf-core-link-format] and can be used by CoAP [I-D.ietf-core-coap] or HTTP servers. The CoRE Link Format defines an attribute that can be used to describe the REST interface of a resource, and may include a link to a description document. This document defines well-known interface descriptions for Batch, Sensor, Parameter, Read-only Parameter and Actuator types for use in contrained web servers. A short reference is provided for each type that can be efficiently included in the interface description (if=) attribute of the CoRE Link Format.

## 2. Terminology

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

This specification requires readers to be familiar with all the terms and concepts that are discussed in [RFC5988] and [I-D.ietf-core-link-format].

## 3. Interface Descriptions

This section defines REST interfaces for Batch, Sensor, Parameter and Actuator resources. Each type is described along with its Interface Description attribute value, recommended path pattern and valid methods. These are defined for each interface in the table below. These interfaces can support plain text and/or SenML Media types.

The if= column defines the Interface Description (if=) attribute value to be used in the CoRE Link Format for a resource conforming to

that interface. When this value appears in the if= attribute of a link, the resource MUST support the corresponding REST interface described in this section. The resource MAY support additional functionality, which is out of scope for this specification. Although these interface descriptions are intended to be used with the CoRE Link Format, they are applicable for use in any REST interface definition.

The Path column defines an example path pattern that a resource of this type might use for simple end-points. These interfaces are expected to also be used in other path patterns. In any case, resource paths SHOULD be discoverable as described in [I-D.ietf-core-link-format].

The Methods column defines the methods supported by that interface, which are described in more detail below.

| Interface | if= | Path Example | Methods |
|---|---|---|---|
| Batch | core#b | /s,/p,/a | GET, PUT, POST (where applicable) |
| Sensor | core#s | /s/{name} | GET |
| Parameter | core#p | /p/{name} | GET, PUT |
| Read-only Parameter | core#rp | /p/{name} | GET |
| Actuator | core#a | /a/{name} | GET, PUT, POST |

The following is an example of links in the CoRE Link Format using these interface descriptions. These links are used in the subsequent examples below.

```
Req: GET /.well-known/core
Res: 2.05 Content (application/link-format)
</s>;if="core#b",
</s/light>;if="core#s",
</s/temp>;if="core#s";obs,
</s/humidity>;if="core#s",
</p/name>;if="core#p",
</p/model>;if="core#rp",
</a>;if="core#b",
</a/led1>;if="core#a",
</a/led2>;if="core#a"
```

### 3.1. Batch

The Batch interface is used to manipulate a collection of sub-resources at the same time. The Batch interface type supports the same methods as its sub-resources, and can be used to read (GET) or set (PUT) or toggle (POST) the values of those sub-resource with a single resource representation. The sub-resources of a Batch MAY be heterogeneous, a method used on the Batch only applies to sub-resources that support it. For example Sensor interfaces do not support PUT, and thus a PUT request to a Sensor member of that Batch would be ignored. A batch requires the use of SenML Media types in order to support multiple sub-resources.

The following example interacts with a Batch /s with Sensor sub-resources /s/light, /s/temp and /s/humidity.

```
Req: GET /s
Res: 2.05 Content (application/senml+json)
{"e":[
    { "n": "light", "v": 123, "u": "lx" },
    { "n": "temp", "v": 27.2, "u": "degC" },
    { "n": "humidity", "v": 80, "u": "%RH" }],
}
```

### 3.2. Sensor

The Sensor interface allows the value of a sensor resource to be read (GET). The Media type of the resource can be either plain text or SenML. Plain text MAY be used for a single measurement that does not require meta-data. For a measurement with meta-data such as a unit or time stamp, SenML SHOULD be used. A resource with this interface MAY use SenML to return multiple measurements in the same representation, for example a list of recent measurements.

The following are examples of Sensor interface requests in both text/plain and application/senml+json.

```
Req: GET /s/humidity (Accept: text/plain)
Res: 2.05 Content (text/plain)
80

Req: GET /s/humidity (Accept: application/senml+json)
Res: 2.05 Content (application/senml+json)
{"e":[
    { "n": "humidity", "v": 80, "u": "%RH" }],
```

```
}
```

## 3.3. Parameter

The Parameter interface allows configurable parameters and other information to be modeled as a resource. The value of the parameter can be read (GET) or set (PUT). Plain text or SenML Media types MAY be returned from this type of interface.

The following example shows request for reading and setting a parameter.

```
Req: GET /p/name
Res: 2.05 Content (text/plain)
node5

Req: PUT /p/name (text/plain)
outdoor
Res: 2.04 Changed
```

## 3.4. Read-only Parameter

The Read-only Parameter interface allows configuration parameters to be read (GET) but not set. Plain text or SenML Media types MAY be returned from this type of interface.

The following example shows request for reading such a parameter.

```
Req: GET /p/model
Res: 2.05 Content (text/plain)
SuperNode200
```

## 3.5. Actuator

The Actuator interface is used by resources that model different kinds of actuators (changing its value has an effect on its environment). Examples of actuators include for example LEDs, relays, motor controllers and light dimmers. The current value of the actuator can be read (GET) or a new actuator value set (PUT). In addition, this interface defines the use of POST (with no body) to toggle an actuator between its possible values. Plain text or SenML Media types MAY be returned from this type of interface. A resource with this interface MAY use SenML to include multiple measurements in

the same representation, for example a list of recent actuator values or a list of values to set.

The following example shows request for reading, setting and toggling an actuator (turning on a led).

```
Req: GET /a/led1
Res: 2.05 Content (text/plain)
0

Req: PUT /a/led1 (text/plain)
1
Res: 2.04 Changed

Req: POST /a/led1 (text/plain)
Res: 2.04 Changed

Req: GET /a/led1
Res: 2.05 Content (text/plain)
0
```

## 3.6. Resource Observation

When resource interfaces following this specification are made available over CoAP, the CoAP Observation mechanism [I-D.ietf-core-observe] MAY be used to observe any changes in a resource, and receive asynchronous notifications as a result. In addition, a set of query string parameters are defined here to allow a client to request how often a client is interested in receiving notifications and how much a resource should change for the new representation to be interesting. These query parameters are described in the following table. A resource using an interface description defined in this specification and marked as Observable in its link description SHOULD support these observation parameters. The Change Step parameter can only be supported on resources with an atomic numeric value.

| Query | Parameter | Value |
|---|---|---|
| Minimum Period (s) | pmin | xsd:integer (>0) |
| Maximum Period (s) | pmax | xsd:integer (>0) |
| Change Step | st | xsd:decimal (>0) |

Minimum Period: When present, the minimum period indicates the minimum time in seconds the server SHOULD wait between sending notifications. In the absence of this parameter, the minimum period is up to the server.

Maximum Period: When present, the maximum period indicated the maximum time in seconds the server SHOULD wait between sending notifications (regardless if it has changed). In the absence of this parameter, the maximum period is up to the server. The maximum period MUST be great than the minimum period parameter (if present).

Change Step: When present, the change step indicates how much the value of a resource SHOULD change before sending a new notification (compared to the value of the last notification). This parameter has lower priority than the period parameters, thus even if the change step has been fulfilled, the time since the last notification SHOULD be between pmin and pmax.

The following example shows an Observation request using these query parameters. Here the value of Observe indicates the number of seconds since the observation was made to show the time.

```
Req: GET Observe /s/temp?pmin=10&pmax=60&st=1
Res: 2.05 Content Observe:0 (text/plain)
23.2

Res: 2.05 Content Observe:60 (text/plain)
23.0

Res: 2.05 Content Observe:80 (text/plain)
22.2

Res: 2.05 Content Observe:140 (text/plain)
21.8
```

## 3.7. Future Interfaces

It is expected that further interface descriptions will be defined in this and other specifications. Potential interfaces to be considered for this specifications include:

Batch of Links: This resource would contain a list of links, pointing to resources that will be manipulated by requesting a method on that resource.

Collection: This resource would be a container that allows sub-resources to be added or removed.

## 3.8. WADL Description

This section defines the formal Web Application Description Langauge (WADL) definition of these CoRE interface descriptions.

```
<?xml version="1.0" standalone="yes"?>

<application xmlns="http://research.sun.com/wadl/2006/10"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<grammars>
<include href="http://tools.ietf.org/html/draft-jennings-senml"/>
</grammars>

<doc title="CoRE Interfaces"/>

<resource_type id="b">
 <doc title="Batch of sub-resources type. The method is applied
 to each sub-resource of the requested resource that
 supports it. Mixed sub-resource types can be supported."/>
 <method id="b_request" name="GET">
    <doc>Retrieve the representations of all resources under
    this collection in a single request.</doc>
    <request>
        <param name="pmin" style="query" type="xsd:integer"/>
        <param name="pmax" style="query" type="xsd:integer"/>
        <param name="st" style="query" type="xsd:decimal"/>
    </request>
    <response status="200">
      <representation mediaType="application/senml+exi"/>
      <representation mediaType="application/senml+xml"/>
      <representation mediaType="application/senml+json"/>
    </response>
 </method>
<method id="b_update" name="PUT">
    <doc>Update the representations of all resources under
    this collection in a single request that support
    PUT.</doc>
    <request>
       <representation mediaType="application/senml+exi"/>
       <representation mediaType="application/senml+xml"/>
       <representation mediaType="application/senml+json"/>
    </request>
    <response status="204"/>
```

```
    </method>
    <method id="b_toggle" name="POST">
       <doc>Toggle the values of actuator resources that support
       POST in the collection.</doc>
       <request>
       </request>
       <response status="204"/>
    </method>
   </resource_type>

   <resource_type id="s">
    <doc title="Sensor resource type"/>
    <method id="s_request" name="GET">
       <doc>Retrieve the value of the sensor</doc>
       <request>
           <param name="pmin" style="query" type="xsd:integer"/>
           <param name="pmax" style="query" type="xsd:integer"/>
           <param name="st" style="query" type="xsd:decimal"/>
       </request>
       <response status="200">
          <representation mediaType="text/plain"/>
          <representation mediaType="application/senml+exi"/>
          <representation mediaType="application/senml+xml"/>
          <representation mediaType="application/senml+json"/>
       </response>
    </method>
   </resource_type>

   <resource_type id="p">
    <doc title="Parameter resource type"/>
    <method id="p_request" name="GET">
       <doc>Retrieve the value of the parameter</doc>
       <request>
           <param name="pmin" style="query" type="xsd:integer"/>
           <param name="pmax" style="query" type="xsd:integer"/>
           <param name="st" style="query" type="xsd:decimal"/>
       </request>
       <response status="200">
          <representation mediaType="text/plain"/>
          <representation mediaType="application/senml+exi"/>
          <representation mediaType="application/senml+xml"/>
          <representation mediaType="application/senml+json"/>
       </response>
    </method>
    <method id="p_update" name="PUT">
       <doc>Update to the parameter value</doc>
       <request>
          <representation mediaType="text/plain"/>
```

```
         <representation mediaType="application/senml+exi"/>
         <representation mediaType="application/senml+xml"/>
         <representation mediaType="application/senml+json"/>
      </request>
      <response status="204"/>
    </method>
   </resource_type>

   <resource_type id="rp">
    <doc title="Read-only Parameter resource type"/>
    <method id="p_request" name="GET">
      <doc>Retrieve the value of the parameter</doc>
      <request>
          <param name="pmin" style="query" type="xsd:integer"/>
          <param name="pmax" style="query" type="xsd:integer"/>
          <param name="st" style="query" type="xsd:decimal"/>
      </request>
      <response status="200">
         <representation mediaType="text/plain"/>
         <representation mediaType="application/senml+exi"/>
         <representation mediaType="application/senml+xml"/>
         <representation mediaType="application/senml+json"/>
      </response>
    </method>
   </resource_type>

   <resource_type id="a">
    <doc title="Actuator resource type"/>
    <method id="a_request" name="GET">
      <doc>Retrieve the value of the actuator</doc>
      <request>
          <param name="pmin" style="query" type="xsd:integer"/>
          <param name="pmax" style="query" type="xsd:integer"/>
          <param name="st" style="query" type="xsd:decimal"/>
      </request>
      <response status="200">
         <representation mediaType="text/plain"/>
         <representation mediaType="application/senml+exi"/>
         <representation mediaType="application/senml+xml"/>
         <representation mediaType="application/senml+json"/>
      </response>
    </method>
    <method id="a_actuate" name="PUT">
      <doc>Control the actuator with a new value or command</doc>
      <request>
         <representation mediaType="text/plain"/>
         <representation mediaType="application/senml+exi"/>
         <representation mediaType="application/senml+xml"/>
```

```
        <representation mediaType="application/senml+json"/>
      </request>
      <response status="204"/>
   </method>
   <method id="a_toggle" name="POST">
      <doc>Toggle the value of the actuator</doc>
      <request>
      </request>
      <response status="204"/>
   </method>
  </resource_type>

  </application>
```

## 4. Security Considerations

An implementation of a client needs to be prepared to deal with responses to a request that differ from what is specified in this document. A server implementing what the client thinks is a resource with one of these interface descriptions could return malformed representations and response codes either by accident or maliciously. A server sending maliciously malformed responses could attempt to take advantage of a poorly implemented client for example to crash the node or perform denial of service.

## 5. IANA Considerations

To be determined if a registry of interface descriptions should be created for CoRE, allowing other interface descriptions to be registered by other specifications (and if this document is the place to create such a registry).

## 6. Acknowledgments

Acknowledgement is given to colleagues from the SENSEI project who were critical in the initial development of the well-known REST interface conept, to members of the IPSO Alliance where further requirements for interface types have been discussed, and to Szymon Sasin, Cedric Chauvenet, Daniel Gavelle and Carsten Bormann who have provided useful discussion and input to the concepts in this document.

Special thanks to Matthieu Vial for helping develop ideas for the batch, link batch and collection resources and for providing useful

comments and the document.

## 7. Changelog

## 8. References

### 8.1. Normative References

[I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format", draft-ietf-core-link-format-11 (work in progress), January 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

### 8.2. Informative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-08 (work in progress), October 2011.

[I-D.ietf-core-observe]
Hartke, K. and Z. Shelby, "Observing Resources in CoAP", draft-ietf-core-observe-03 (work in progress), October 2011.

## Author's Address

Zach Shelby
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com