Constrained RESTful Environments	M.V. Vial
Internet-Draft	Schneider-Electric
Intended status: Informational	Z. Shelby
	Sensinode
	Sept 2011

Interface description with WADL in CoRE
draft-vial-core-link-format-wadl-01

Abstract

This document provides guidelines to use the Web Application Description Language (WADL) in Constrained RESTful environments. The document mainly focuses on how to combine WADL with the CoRE Link Format to describe a REST interface.

Status of this Memo

Copyright Notice

Table of Contents

*1. Introduction

*2. Requirements Language

*3. WADL in a CoRE environment

*3.1. CoAP adaptations

*3.1.1. Methods

*3.1.2. Status code

*3.1.3. HTTP header parameters

*3.2. CoRE resources

*3.3. Semantic description

*3.4. Binary XML format

*4. Use cases

*4.1. WADL resource type identifiers

*4.2. Description of query parameters

*4.3. Interface description and associated semantic

*4.4. Collection of resources

*5. Acknowledgements

*6. IANA Considerations

*7. Security Considerations

*8. References

*8.1. Normative References

*8.2. Informative References

*Authors' Addresses

1. Introduction

The Constrained RESTful Environments (CoRE) working group aims at providing a comprehensive suite of standards that will make it possible to build a REST architecture for M2M applications with highly constrained nodes and networks.
The CoRE Link Format *[I-D.ietf-core-link-format]* which is part of this suite defines a format to be used by CoAP servers to list hosted resources using the Web linking technique defined in RFC 5988 *[RFC5988]*. More specically the 'if' attribute of Link Format allows an interface designer indicate a description of the behavior, the parameters, the representation and eventually the set of sub-resources

associated to a given CoRE resource. One way to describe the interface to a resource is using the Web Application Description Language (WADL). The first part of this document will explain how to benefit from WADL to describe the REST interface of CoRE resources. Then the second part of the document will show how the previous guidelines are applied in different use cases.
The reader should keep in mind that this document does not suggest in any way constrained nodes would be able to retrieve and parse a WADL description. The interface description is rather considered as documentation with a standard and machine-processable language that will help implementors to understand an interface and eventually generate stub code.

2. Requirements Language

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 RFC 2119 *[RFC2119]*.

3. WADL in a CoRE environment

3.1. CoAP adaptations

The WADL language is primilarily designed to describe HTTP-based web applications. WADL is not strongly tied to the HTTP protocol *[RFC2616]* and any HTTP-like web protocol can be described with WADL. In a CoRE environment the CoAP protocol *[I-D.ietf-core-coap]* is deployed in place of the HTTP protocol as an optimized web transfer protocol. An interface designer must take into consideration the specifity of CoAP while writing the WADL definition.

3.1.1. Methods

CoAP only supports a subset of HTTP methods. So a WADL description deployed in a CoRE environment MUST only make use of methods available in CoAP namely GET, PUT, POST and DELETE.

3.1.2. Status code

CoAP decorrelates the response code representation from the actual value of the code. Hence the response code 2.00 has the value 64. When a CoAP response code is associated to the description of a response in WADL, it is RECOMMENDED to use the response code labels.

3.1.3. HTTP header parameters

CoAP does not support user-defined options in the base specification. So as a rule of thumb, header parameters are discouraged with CoAP.

3.2. CoRE resources

The WADL language describes a REST resource with a resource element which associates a REST behavior to a URI. WADL offers language elements to describe the following aspects of a REST behavior: allowed methods, query string parameters, media type of the request and response content, URI of the resource and the subordinate resources. The description of the behavior can either be a reference to a resource_type element or child elements if the description is inline. When WADL is combined with the CoRE Link Format it is RECOMMENDED to write definitions with resource_type elements. Then a Web link can reference a resource_type with the Interface Description 'if' attribute. So a Web link plays the same role as the resource element of WADL in the sense that the Web link instantiates a resource_type by linking it to an URI.
Because the resource_type element is referenced outside of the WADL description, the rules of section 2.5.1 in WADL *[wadl]* are not applicable. Instead the target URI of the Web link where the resource_type element is referenced MUST be used as the base URI to construct each child resource identifiers.
The 'if' attribute of a Web link SHOULD reference a resource_type AND a WADL document to avoid potential ambiguities. resource_type elements are identified by their XML id. The 'if' attribute MAY take the form of an URI. The path of the URI specify the WADL document while the fragment part of the URI points to a resource_type. The full URI notation may add significant overhead in a link format description thus several formats are acceptable depending on the risk of identifier conflicts. Here are few examples of 'if' attributes.

*http://www.example.org/interface.wadl#resourceType

*interface.wadl#resourceType

*resourceType

3.3. Semantic description

The main goal of the WADL description in a CoRE environment is to describe the actions that can be performed on a REST resource. The WADL document may include a grammar element with a schema to offer a detailed description of data representations hence semantically refining the description. But this mechanism is not applicable to all data representations especially if the data is not XML-based. Moreover the interface description is not meant to be directly interpreted by CoRE nodes. Thus it is RECOMMENDED to associate the semantic description of a resource with a resource type 'rt' attribute without relying only on the 'if' attribute. This separation of concerns allows an interface designer to reuse the same interface description for resources that grasp different concepts.

3.4. Binary XML format

Because CoRE deals with constrained networks, traditional XML data representations may be superseded with a more compact format for the XML information set. Efficient XML Interchange *[exi]* is an example of such binary XML format which heavily relies on a XML schema to achieve the best compression performances. The schema identifier can be carried inline with the binary stream or specified out-of-band. If there is no schema identifier present in the data stream but the WADL definition of the representation has a reference to grammar element, one MUST assume that the data stream is schema-informed.

4. Use cases

4.1. WADL resource type identifiers

Let's consider an organization which has defined two application profiles in two separate WADL documents. The first profile targets Home Automation applications while the second deals with Energy Management. One CoRE device may implement REST services from both profiles.
The WADL descriptions are versioned to support future evolutions of the interface. The profiles have a class/type structure with a dot notation for REST resource types. They also share some concepts but with different implementations. Figure 1 and Figure 2 are short extracts of possible WADL descriptions for such profiles.

```
<application xmlns="http://wadl.dev.java.net/2009/02"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd">
  <resource_type id="sensor.temperature">
    <method name="GET">
      <doc>GetTemperature</doc>
      <response>
        <representation mediaType="text/plain" />
      </response>
    </method>
  </resource_type>
  <resource_type id="parameter.threshold">
    <method name="PUT">
      <doc>SetThreshold</doc>
      <request>
        <representation mediaType="text/plain" />
      </request>
    </method>
 </resource_type>
</application>
```

Home Automation WADL sample (ha1.wadl)

```
<application xmlns="http://wadl.dev.java.net/2009/02"
xmlns:em2="http://www.example.org/EnergyManagement/2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd">
  <grammars>
    <include href="http://www.example.org/EnergyManagement/2/em2.xsd"/>
  </grammars>
  <resource_type id="meter.power">
    <method name="GET">
      <doc>GetPower</doc>
      <response>
      <representation mediaType="application/xml" element="em2:Power"/>
      </response>
    </method>
  </resource_type>
  <resource_type id="parameter.threshold">
    <method name="PUT">
      <doc>SetThreshold</doc>
      <request>
  <representation mediaType="application/xml" element="em2:Threshold"/>
      </request>
    </method>
  </resource_type>
</application>
```

Energy Management WADL sample (em2.wadl)
In a home network, the devices share the same infrastructure but usually come from different vendors and may implement many application profiles. In this context it is useful to reference a WADL interface with an absolute URI.

```
REQ: GET /.well-known/core
RES: 2.00 OK
</tmp>;rt="AirTemperature";
if="http://www.example.org/ha1.wadl#sensor.temperature",
</tmp/thr>;rt="TemperatureAlarm";
if="http://www.example.org/ha1.wadl#parameter.threshold"
</pwr>;rt="PowerConsumption";
if="http://www.example.org/em2.wadl#meter.power",
</pwr/thr>;rt="PowerAlarm";
if="http://www.example.org/em2.wadl#parameter.threshold"
```

If a deployment of devices is known to implement only REST services from one organization the resource_type identifiers may be shortened. It is however indispensable to clearly indicate a WADL document because the resource_type identifiers are only unique within a single WADL document. The example below reflects how these assumptions are actually applied.

```
REQ: GET /.well-known/core
RES: 2.00 OK
</tmp>;rt="AirTemperature";if="ha1.wadl#sensor.temperature",
</tmp/thr>;rt="TemperatureAlarm";if="ha1.wadl#parameter.threshold"
</pwr>;rt="PowerConsumption";if="em2.wadl#meter.power",
</pwr/thr>;rt="PowerAlarm";if="em2.wadl#parameter.threshold"
```

If the network is dedicated to a specific application profile it is acceptable to omit the reference to the WADL description which is supposed to be known out-of-band. Web links may have the following format:

```
REQ: GET /.well-known/core
RES: 2.00 OK
</tmp>;rt="AirTemperature";if="sensor.temperature",
</thr>;rt="TemperatureAlarm";if="parameter.threshold"
```

4.2. Description of query parameters

A typical usage of WADL is to provide a detailed description of how a client can build the query string component of a URI to access a parametrized resource. Below is an example that describes how a client can select the unit for a temperature sensor.

```
<application xmlns="http://wadl.dev.java.net/2009/02"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd">
  <resource_type id="sensor.temperature">
    <method name="GET">
      <doc>GetTemperature</doc>
      <request>
        <param name="unit" style="query" default="C" required="no">
          <option value="C"><doc>Celsius</doc></option>
          <option value="K"><doc>Kelvin</doc></option>
          <option value="F"><doc>Farenheit</doc></option>
        </param>
      </request>
      <response>
        <representation mediaType="text/plain" />
      <response>
    </method>
  </resource_type>
</application>
```

Definition of an optional query string
This description give information about four valid URIs that are exposed in the following CoAP exchange.

```
REQ: GET /.well-known/core
RES: 2.00 OK
</tmp>;if="sensor.temperature"

REQ: GET /tmp
RES: 2.00 OK
20

REQ: GET /tmp?unit=C
RES: 2.00 OK
20

REQ: GET /tmp?unit=K
RES: 2.00 OK
293.15

REQ: GET /tmp?unit=F
RES: 2.00 OK
68
```

4.3. Interface description and associated semantic

The same interface description can often be reused for similar but distinct concepts. For example a temperature sensor may be able to produce the traditional air temperature but also the effective temperature which is a combination of air temperature and wind speed. Then the definition in Figure 6 is valid for both concepts and the device description could look like depicted below.

```
REQ: GET /.well-known/core
RES: 2.00 OK
</tmp>;rt="DryBulbTemperature";if="sensor.temperature",
</eff>;rt="EffectiveTemperature";if="sensor.temperature"
```

4.4. Collection of resources

Repeating an interface definition attribute with the same identifier for a collection of resources is especially inefficient and laborious with link format. Hopefully WADL supports templated path definitions to describe sub-resources. The template style of parameters allows an interface designer to specify the dynamic path elements of a URI thanks to a curly brace notation. It is also possible to precisely determine the data type associated to a variable path element. Below is an

example of how a list of pending alarms can be described with this
feature.

```
<application xmlns="http://wadl.dev.java.net/2009/02"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:app="http://www.example.org/2011/app"
xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd">
  <grammars>
    <include href="http://www.example.org/2011/app/app.xsd"/>
  </grammars>
  <resource_type id="list.alarms">
    <method name="GET">
      <doc>GetAlarmList</doc>
      <response>
        <representation mediaType="application/link-format"/>
      </response>
    </method>
    <method name="POST">
      <doc>AddAlarm</doc>
      <request>
        <representation mediaType="application/xml"
            element="Alarm"/>
      </request>
    </method>
    <resource path="{alarmId}">
      <param name="alarmId" style="template" type="xsd:int"/>
      <method name="GET">
        <doc>GetAlarm</doc>
        <response>
          <representation mediaType="application/xml"
              element="app:Alarm"/>
        </response>
      </method>
      <method name="DELETE">
        <doc>RemoveAlarm</doc>
        <request>
        </request>
      </method>
    </resource>
  </resource_type>
</application>
```

Definition of a collection of resources
Then the resource_type is referenced only once but provides an
interface description for the whole collection of resources.

```
REQ: GET /.well-known/core
RES: 2.00 OK
</tmp>;rt="DryBulbTemperature";if="sensor.temperature",
</alrm>;rt="TemperatureAlarms";if="list.alarms",

REQ: GET /alrm
RES: 2.00 OK
</alrm/1>,
</alrm/2>

REQ: GET /alrm/1
RES: 2.00 OK
<Alarm time="" type="GreaterThan" threshold="28" />
```

5. Acknowledgements

Thanks to Linyi Tian for its feedback on the document.

6. IANA Considerations

This document requests no actions from IANA.

7. Security Considerations

This document has no known security implications.

8. References

8.1. Normative References

[wadl]	Hadley, M.J.H, "Web Application Description Language (WADL)", 2009.
[I-D.ietf-core-link-format]	Shelby, Z, "CoRE Link Format", Internet-Draft draft-ietf-core-link-format-09, November 2011.
[I-D.ietf-core-coap]	Shelby, Z, Hartke, K, Bormann, C and B Frank, "Constrained Application Protocol (CoAP)", Internet-Draft draft-ietf-core-coap-08, October 2011.
[RFC5988]	Nottingham, M., "Web Linking", RFC 5988, October 2010.
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[exi]	W3C, "Efficient XML Interchange (EXI) Format 1.0", 2011.

8.2. Informative References

[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

Authors' Addresses

Matthieu Vial Vial Schneider-Electric Grenoble, FRANCE Phone: +33 (0)47657 6522 EMail: matthieu.vial@schneider-electric.com

Zach Shelby Shelby Sensinode Kidekuja 2 Vuokatti, 88600 FINLAND Phone: +358407796297 EMail: zach@sensinode.com