CoRE
Internet-Draft
Intended status: Standards Track
Expires: March 18, 2012

Z. Shelby
Sensinode
S. Krco
Ericsson

September 15, 2011

CoRE Resource Directory
draft-shelby-core-resource-directory-01

Abstract

In many M2M scenarios, direct discovery of resources is not practical due to sleeping nodes, disperse networks, or networks where multicast traffic is inefficient. These problems can be solved by employing an entity called a Resource Directory (RD), which hosts descriptions of resources held on other servers, allowing lookups to be performed for those resources. This document specifies the web interfaces that a Resource Directory supports in order for web servers to discover the RD and to registrer, maintain, lookup and remove resources descriptions. Furthermore, new link attributes useful in conjunction with an RD are defined.

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 March 18, 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 [I-D.shelby-core-coap-req].

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]. This specification however only describes how to discover resources from the web server that hosts them by requesting /.well-known/core. In many M2M scenarios, direct discovery of resources is not practical due to sleeping nodes, disperse networks, or networks where multicast traffic is inefficient. These problems can be solved by employing an entity called a Resource Directory (RD), which hosts descriptions of resources held on other servers, allowing lookups to be performed for those resources.

This document specifies the web interfaces that a Resource Directory supports in order for web servers to discover the RD and to registrer, maintain, lookup and remove resources descriptions. Furthermore, new link attributes useful in conjunction with a Resource Directory are defined. Although the examples in this document show the use of these interfaces with CoAP [I-D.ietf-core-coap], they may be applied in an equivalent manner to HTTP [RFC2616].

2. Architecture and Use Cases

The resource directory architecture is shown in Figure 1. A Resource Directory (RD) is used as a repository for Web Links [RFC5988] about resources hosted on other web servers, which are called end-points (EP). An end-point is a web server associated with a port, thus a physical node may host one or more end-points. The RD implements a set of REST interfaces for end-points to register and maintain sets of Web Links (called resource directory entries), for the RD to validate entries, and for clients to lookup resources from the RD. End-points themselves can also act as clients.

End-points are assumed to proactively register and maintain resource directory entries on the RD, which are soft state and need to be periodially refreshed. An EP is provided with interfaces to register, update and remove a resource directory entry. Furthermore, a mechanism to discover a RD using the CoRE Link Format is defined. It is also possible for an RD to proactively discover Web Links from EPs and add them as resource directory entries, or to validate existing resource directory entries. A lookup interface for discovering any of the Web Links held in the RD is provided using the CoRE Link Format.

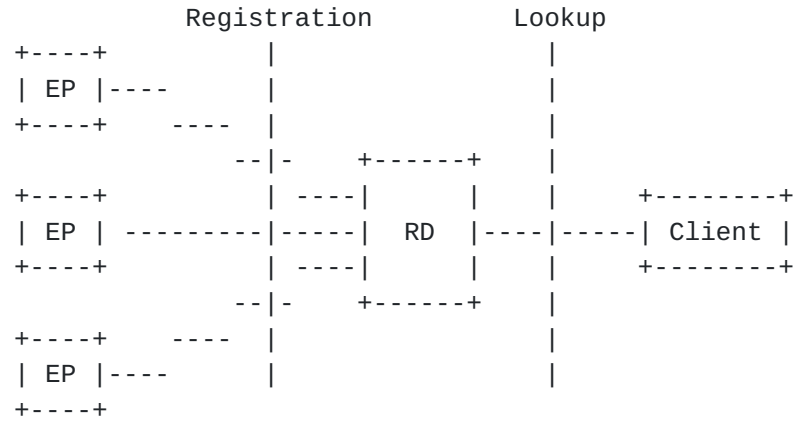


Figure 1: The resource directory architecture.

2.1. Use Case: Cellular M2M

Over the last few years, mobile operators around the world have focused on development of M2M solutions in order to expand the business to the new type of users, i.e. machines. The machines are connected directly to a mobile network using appropriate embedded air interface (GSM/GPRS, WCDMA, LTE) or via a gateway providing short and wide range wireless interfaces. From the system design point of view, the ambition is to design horizontal solutions that can enable utilization of machines in different applications depending on their current availability and capabilities as well as application requirements, thus avoiding silo like solutions. One of the crucial enablers of such design is the ability to discover resources (machines - End Points) capable of providing required information at a given time or acting on instructions from the end users.

In a typical scenario, during a boot-up procedure (and periodically afterwards), the machines (EPs) register with a Resource Directory (for example EPs installed on vehicles enabling tracking of their position for the fleet management purposes and monitoring environment

parameters) hosted by the mobile operator or somewhere else in the network, submiting a description of own capabilities. Due to the usual network configuration of mobile networks, the EPs attached to the mobile network do not have routable addresses. Therefore, a remote server is usually used to provide proxy access to the EPs. The address of each (proxy) EP on this server is included in the resource description stored in the RD. The users, for example mobile applications for environment monitoring, contact the RD, look-up the EPs capable of providing information about the environment using appropriate set of tags, obtain information on how to contact them (URLs of the proxy server) and then initate interaction to obtain information that is finally processed, displayed on the screen and usually stored in a database. Similarly, fleet management systems provide a set of credentials along with the appropriate tags to the RD to look-up for EPs deployed on the vehicles the application is responsible for.

2.2. Use Case: Home and Building Automation

Home and commercial building automation systems can benefit from the use of M2M web services. The use of CoRE in home automation across multiple subnets is described in [I-D.brandt-coap-subnet-discovery] and in commercial building automation in [I-D.vanderstok-core-bc]. The discovery requirements of these applications are demanding. Home automation usually relies on run-time discovery to commision the system, whereas in building automation a combination of professional commissioning and run-time discovery. Both home and building automation involve peer-to-peer interactions between end-points, and involve battery-powered sleeping devices.

The exporting of resource information to other discovery systems is also important in these automation applications. In home automation there is a need to interact with other consumer electronics, which may already support DNS-SD, and in building automation larger resource directories or DNS-SD covering multiple buildings.

3. Resource Directory Interfaces

This section defines the REST interfaces between an RD and end-points, and a lookup interface between an RD and clients. Although the examples throughout this section assume use of CoAP [I-D.ietf-core-coap], these REST interfaces can also be realized using HTTP [RFC2616]. An RD implementing this specification MUST support the discovery, registration, update, removal and lookup interfaces defined in this section and MAY support the validation interface. For the purpose of validation, an end-point implementing this specification SHOULD support Etag validation on /.well-known/

core.

3.1. Discovery

Before an end-point can make use of an RD, it must first know its location and optionally the path of the RD root resource. There can be several mechanisms for discovering the RD including assuming a default location (e.g. on an Edge Router in a LoWPAN), by assigning an anycast address to the RD, using DHCP, or by discovering the RD using the CoRE Link Format. This section defines discovery of the RD using the well-known interface of the CoRE Link Format [I-D.ietf-core-link-format] as the required mechanism. It is however expected that RDs will also be discoverable via other methods depending on the deployment.

Discovery is performed by sending either a multicast or unicast GET request to /.well-known/core and including a Resource Type (rt) parameter [I-D.ietf-core-link-format] with the value "core-rd" in the query string. Upon success, the response will contain a payload with a link format entry for each RD discovered, with the URL indicating the root resource of the RD. When performing multicast discovery, the multicast IP address used will depend on the scope required and the multicast capabilities of the network (TBD if a specific multicast address should be defined for RDs).

An RD implementing this specification MUST support query filtering for the rt parameter as defined in [I-D.ietf-core-link-format].

The discovery interface is specified as follows:

Interaction: EP -> RD

Path: /.well-known/core

Method: GET

Content-Type: application/link-format (if any)

Parameters:

Resource Type (rt): MUST contain the value "core-rd"

Instance (ins): Used to differentiate between multiple RDs.

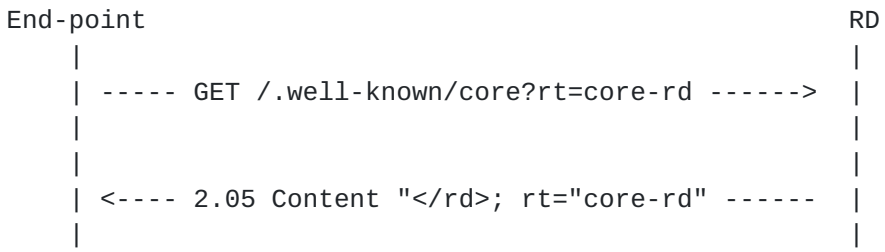
Success: 2.05 "Content" with an application/link-format payload containing a matching entry for the RD resource.

Failure: 2.05 "Content" (should be a "No Content" code?) with an empty payload is returned in case no matching entry is found for a unicast request.

Failure: No error response to a multicast request.

Failure: 4.00 "Bad Request"

The following example shows an end-point discovering an RD using this interface, thus learning that the base RD resource is at /rd. Note that it is up to the RD to choose its base RD resource.



Req: GET coap://[ff02::1]/.well-known/core?rt=core-rd

Res: 2.05 Content
</rd>; rt="core-rd"; ins="Primary"

3.2. Registration

After discovering the location of an RD, an end-point MAY register its resources to the RD's registration interface. This interface accepts a POST from an end-point containing the list of resources to be added to the directory as the message payload in the CoRE Link Format along with query string parameters indicating the name of the end-point, an optional node identifier and the lifetime of the registration. The end-point name is formed by concatenating the Host and Instance parameters included with the registration. All parameters of the registration are optional. In the absense of a Host parameter, the RD will generate a unique one on behalf of the end-point. The RD then creates a new resource or updates an existing resource in the RD and returns its location. An end-point MUST use that location when refreshing registrations using this interface.

End-point resources in the RD are kept active for the period indicated by the lifetime parameter. The end-point is reponsible for refreshing the entry within this period using either the registration or update interface.

The registration interface is specified as follows:

Interaction: EP -> RD

Path: /.well-known/core or /{rd-base}

Method: POST

Content-Type: application/link-format

Etag: The Etag option MUST be included to allow an RD to perform validation in the future.

Parameters:

- Lifetime (lt): Lifetime of the registration in seconds. Range of 60-4294967295. If no lifetime is included, a default value of 86400 (24 hours) SHOULD be assumed.

- Host (h): The host identifier or name of the registering node. The maximum length of this parameter is 63 octets. This parameter is combined with the Instance parameter (if any) to form the end-point name. If not included, the RD MUST generate a unique Host name on behalf of the node.

- Instance (ins): The instance of the end-point on this host, if there are multiple. The maximum length of this parameter is 63 octets.

- Type (rt): The semantic type of end-point. The maximum length of this parameter is 63 octets.

Success: 2.01 "Created". The Location header of the new resource entry for the end-point could be e.g. in the form /{rd-base}/{end-point name}

Failure: 4.00 "Bad Request". Malformed request.

Failure: 5.03 "Service Unavailable". Service could not perform the operation.

The following example shows an end-point with the name "node1" registering two resources to an RD using this interface.


```
Req: POST coap://rd.example.org/rd?h=node1&lt=1024
Etag: 0x3f
Payload:
</sensors/temp>;ct=41;rt="TemperatureC";if="sensor",
</sensors/light>;ct=41;rt="LightLux";if="sensor"

Res: 2.01 Created
Location: /rd/node1
```

3.3. Update

The update interface is used by an end-point to refresh or update its registration with an RD. To use the interface, the end-point sends a PUT request to the resource returned in the Location option in the response to the first registration. An update MAY contain a payload in CoRE Link Format if there have been changes since the last registration or update.

The update interface is specified as follows:

Interaction: EP -> RD

Path: Location returned by registration.

Method: PUT

Content-Type: application/link-format (if any)

Etag: The Etag option MUST be included to allow an RD to compare the existing entry and perform validation in the future.

Parameters:

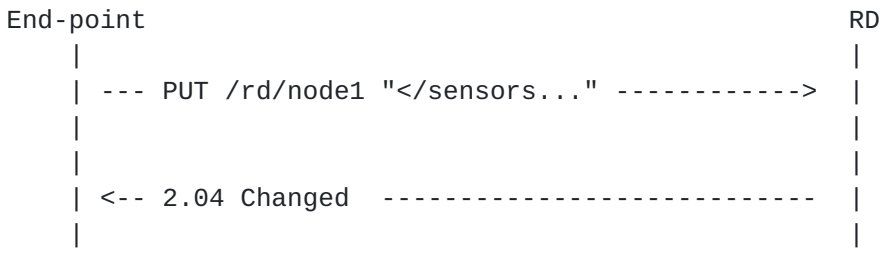
Lifetime (lt): Lifetime of the registration in seconds. Range of 60-4294967295. If no lifetime is included, a default value of 86400 (24 hours) SHOULD be assumed.

Success: 2.04 "Changed" in case the resource and/or lifetime was successfully updated

Failure: 4.00 "Bad Request". Malformed request.

Failure: 5.03 "Service Unavailable". Service could not perform the operation.

The following example shows an end-point updating a new set of resources to an RD using this interface.



Req: PUT /rd/node1
Etag: 0x40
Payload:
</sensors/temp/1>;ct=41;ins="Indoor";rt="TemperatureC";if="sensor",
</sensors/temp/2>;ct=41;ins="Outdoor";rt="TemperatureC";if="sensor",
</sensors/light>;ct=41;rt="LightLux";if="sensor"

Res: 2.04 Changed

3.4. Validation

In some cases, an RD may want to validate that it has the latest version of an end-point's resource. This can be performed with a GET on the well-known interface of the CoRE Link Format including the latest Etag stored for that end-point. For the purpose of validation, an end-point implementing this specification SHOULD support Etag validation on /.well-known/core.

The validation interface is specified as follows:

Interaction: RD -> EP

Path: /.well-known/core

Method: GET

Content-Type: application/link-format (if any)

Etag: The Etag option MUST be included

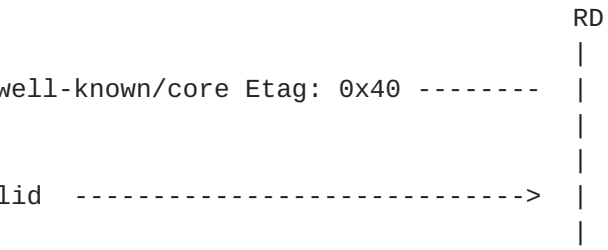
Parameters: None

Success: 2.03 "Valid" in case the Etag matches

Success: 2.05 "Content" in case the Etag does not match, the response MUST include the most recent resource representation and its corresponding Etag.

Failure: 4.00 "Bad Request". Malformed request.

The following examples shows a successful validation.



Req: GET coap://{end-point}/.well-known/core
Etag: 0x40

Res: 2.03 Valid

3.5. Removal

Although RD entries have soft state and will eventually timeout after their lifetime, an end-point SHOULD explicitly remove its entry from the RD if it knows it will no longer be available (for example on shut-down). This is accomplished using a removal interface on the RD by performing a DELETE on the end-point resource.

The removal interface is specified as follows:

Interaction: EP -> RD

Path: Location returned by registration.

Method: DELETE

Content-Type: None

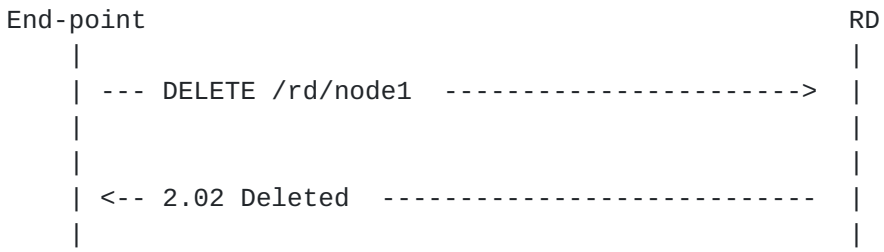
Parameters: None

Success: 2.02 "Deleted" upon successful deletion

Failure: 4.00 "Bad Request". Malformed request.

Failure: 5.03 "Service Unavailable". Service could not perform the operation.

The following examples shows successful removal of the end-point from the RD.



Req: DELETE /rd/node1

Res: 2.02 Deleted

3.6. Lookup

In order for an RD to be used for discovering resources registered with it, a lookup interface is provided. This lookup interface is provided as a default, and it is assumed that RDs may also support lookups to return resource descriptions in alternative formats (e.g. Atom or HTML Link) or using more advanced interfaces (e.g. supporting context or semantic based lookup).

The lookup interface is provided using the CoRE Link Format [I-D.ietf-core-link-format] resource discovery mechanism on the root RD resource (/rd in the examples). The scope of the discovery is controlled by the depth of the resource the query is made on. A lookup on the root RD resource /rd queries all resource descriptions on the RD, whereas a lookup on /rd/node1 queries all resource

descriptions held in the "node1" entry. An RD MUST support the query filtering defined in [I-D.ietf-core-link-format] to allow for filtered lookups.

The lookup interface is specified as follows:

Interaction: Client -> RD

Path: /{rd-base} or e.g. /{rd-base}/{end-point}

Method: GET

Content-Type: application/link-format (if any)

Parameters:

Filtering: CoRE Link Format attributes may be included to filter the lookup.

Success: 2.05 "Content" with an application/link-format payload containing a matching entries for the lookup.

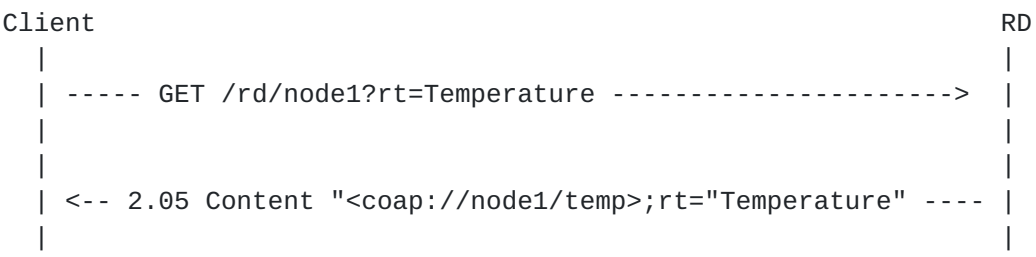
Failure: 2.05 "Content" (should be a "No Content" code?) with an empty payload is returned in case no matching entry is found for a unicast request.

Failure: No error response to a multicast request.

Failure: 4.00 "Bad Request". Malformed request.

Failure: 5.03 "Service Unavailable". Service could not perform the operation.

The following example shows a client performing a lookup on an RD using this interface.



```
Req: GET /rd/node1?rt=Temperature

Res: 2.05 Content
<coap://node1/temp>;rt="Temperature"
```

4. New Link-Format Attributes

When using the CoRE Link Format to describe resources being discovered by or posted to a resource directory service, additional information about those resources is useful. This specification defines the following new attributes for use in the CoRE Link Format [I-D.ietf-core-link-format]:

```
link-extension    = ( "ins" "=" quoted-string ) ; Max 63 octets
link-extension    = ( "exp" )
```

4.1. Resource Instance 'ins' attribute

The Resource Instance "ins" attribute is an identifier for this resource, which makes it possible to distinguish from other similar resources. This attribute is similar in use to the "Instance" portion of a DNS-SD record, and SHOULD be unique across resources with the same Resource Type attribute in the domain it is used. A Resource Instance might be a descriptive string like "Ceiling Light, Room 3", a short ID like "AF39" or a unique UUID or iNumber. This attribute is used by a Resource Directory to distinguish between multiple instances of the same resource type within a system.

This attribute MUST be no more than 63 octets in length. The resource identifier attribute MUST NOT appear more than once in a link description.

4.2. Export 'exp' attribute

The Export "exp" attribute is used as a flag to indicate that a link description MAY be exported by a resource directory to external directories.

The CoRE Link Format is used for many purposes between CoAP end-points. Some are useful mainly locally, for example checking the observability of a resource before accessing it, determining the size of a resource, or traversing dynamic resource structures. However, other links are very useful to be exported to other directories, for example the entry point resource to a functional service.

5. Security Considerations

This document needs the same security considerations as described in Section 7 of [RFC5988] and Section 6 of [I-D.ietf-core-link-format]. The /.well-known/core resource may be protected e.g. using DTLS when hosted on a CoAP server as described in [I-D.ietf-core-coap].

6. IANA Considerations

"core-rd" resource type needs to be registered if an appropriate registry is created.

"ins" and "exp" attributes need to be registered when a future Web Linking attribute is created.

7. Acknowledgments

Szymon Sasin, Carsten Bormann, Kerry Lynn, Peter van der Stok, Anders Brandt and Linyi Tian have provided helpful comments, discussions and ideas to improve and shape this document. The authors would also like to thank their collagues from the EU FP7 SENSEI project, where many of the resource directory concepts were originally developed.

8. Changelog

9. References

9.1. Normative References

[I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format", draft-ietf-core-link-format-06 (work in progress), June 2011.

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

9.2. Informative References

[I-D.brandt-coap-subnet-discovery]
Brandt, A., "Discovery of CoAP servers across subnets", draft-brandt-coap-subnet-discovery-00 (work in progress), March 2011.

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-06 (work in progress), May 2011.

[I-D.shelby-core-coap-req] Shelby, Z., Stuber, M., Sturek, D., Frank, B., and R. Kelsey, "CoAP Requirements and Features", draft-shelby-core-coap-req-01 (work in progress), April 2010.

[I-D.vanderstok-core-bc] Stok, P. and K. Lynn, "CoAP Utilization for Building Control", draft-vanderstok-core-bc-03 (work in progress), March 2011.

[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

Zach Shelby
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

Srdjan Krco
Ericsson

Phone:
Email: srdjan.krco@ericsson.com