                          CoRE Resource Directory
                  draft-shelby-core-resource-directory-05

Abstract

   In many M2M applications, 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 register, 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 August 29, 2013.

Copyright Notice

Table of Contents

1. Introduction

   The Constrained RESTful Environments (CoRE) work 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 [RFC6690]. 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 resource 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. 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]. The term
   "byte" is used in its now customary sense as a synonym for "octet".

   This specification requires readers to be familiar with all the terms
   and concepts that are discussed in [RFC5988] and [RFC6690]. Readers
   should also be familiar with the terms and concepts discussed in
   [I-D.ietf-core-coap]. The URI Template format is used to describe
   the REST interfaces defined in this specification [RFC6570]. This
   specification makes use of the following additional terminology:

Resource Directory
    An web entity that stores information about web resources and
    implements the REST interfaces defined in this specification for
    registration and lookup of those resources.

Domain
    In the context of a Resource Directory, a domain is a logical
    grouping of endpoints. All endpoint within a domain MUST be
    unique. This specification assumes that the list of Domains
    supported by an RD is pre-configured by that RD.

Group
    In the context of a Resource Directory, a group is a logical
    grouping of endpoints for the purpose of group communications.
    All groups within a domain MUST be unique.

Endpoint
    An endpoint (EP) is a term used to describe a web server or client
    in [I-D.ietf-core-coap]. In the context of this specification an
    endpoint is used to describe a web server that registers resources
    to the Resource Directory. An endpoint is identified by its
    endpoint name, which is included during registration, and MUST be
    unique within the associated domain of the registration.

3. 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 endpoints
   (EP). An endpoint is a web server associated with a port, thus a
   physical node may host one or more endpoints. The RD implements a
   set of REST interfaces for endpoints 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. Endpoints
   themselves can also act as clients. An RD can be logically segmented
   by the use of Domains. The domain an endpoint is associated with can
   be defined by the RD or configured by an outside entity.

   Endpoints are assumed to proactively register and maintain resource
   directory entries on the RD, which are soft state and need to be
   periodically refreshed. An endpoint 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
   endpoints 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.

```
              Registration         Lookup
  +----+          |                |
  | EP |----      |                |
  +----+    ----  |                |
              --|-     +------+    |
  +----+        | ----|      |    |    +--------+
  | EP | --------|-----|  RD  |----|-----| Client |
  +----+        | ----|      |    |    +--------+
              --|-     +------+    |
  +----+    ----  |                |
  | EP |----      |                |
  +----+
```
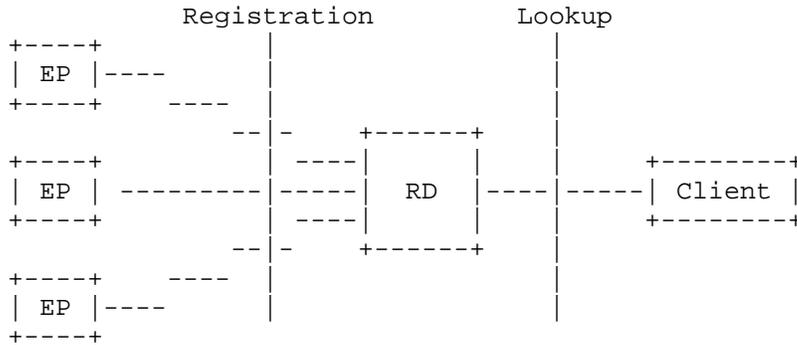
Figure 1: The resource directory architecture.

3.1. 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 - endpoints) 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 (endpoints) 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) endpoint 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 endpoints 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.

3.2. 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 is used. Both home and building
automation involve peer-to-peer interactions between endpoints, 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.

4. Simple Directory Discovery

Not all endpoints hosting resources are expected to know how to
implement the Resource Directory Function Set and thus explicitly
register with a Resource Directory (or other such directory server).
Instead, simple endpoints can implement the generic Simple Directory
Discovery approach described in this section. An RD implementing
this specification MUST implement Simple Directory Discovery.
However, there may be security reasons why this form of directory
discovery would be disabled.

This approach requires that the endpoint makes the hosted resources
that it wants discovered available as links on its /.well-known/core
interface as specified in [RFC6690].

The endpoint then finds one or more IP addresses of the directory
server it wants to know about its resources as described in
Section 4.1.

An endpoint that wants to make itself discoverable occasionally sends

a POST request to the /.well-known/core URI of any candidate
directory server that it finds. The body of the POST request is
either

o empty, in which case the directory server is encouraged by this
   POST request to perform GET requests at the requesting server's
   default discovery URI.

or

o a link-format document, which indicates the specific services that
   the requesting server wants to make known to the directory server.

The directory server integrates the information it received this way
into its resource directory. It MAY make the information available
to further directories, if it can ensure that a loop does not form.
The protocol used between directories to ensure loop-free operation
is outside the scope of this document.

The following example shows an endpoint using simple resource
discovery, by simply sending a POST with its links in the body to a
directory.

```
      EP                                                      RD
       |                                                       |
       | -- POST /.well-known/core "</sen/temp>..." --->       |
       |                                                       |
       |                                                       |
       | <---- 2.01 Created   ------------------------         |
       |                                                       |
```

4.1. Finding a Directory Server

   Endpoints that want to contact a directory server can obtain
   candidate IP addresses for such servers in a number of ways.

   In a 6LoWPAN, good candidates can be taken from:

   o specific static configuration (e.g., anycast addresses), if any,

   o the ABRO option of 6LoWPAN-ND [RFC6775],

   o other ND options that happen to point to servers (such as RDNSS),

   o DHCPv6 options that might be defined later.

   In networks with more inexpensive use of multicast, the candidate IP
   address may be a well-known multicast address, i.e. directory servers
   are found by simply sending POST requests to that well-known
   multicast address (details TBD).

   As some of these sources are just (more or less educated) guesses,
   endpoints MUST make use of any error messages to very strictly rate-
   limit requests to candidate IP addresses that don't work out. E.g.,
   an ICMP Destination Unreachable message (and, in particular, the port
   unreachable code for this message) may indicate the lack of a CoAP
   server on the candidate host, or a CoAP error response code such as
   4.05 "Method Not Allowed" may indicate unwillingness of a CoAP server
   to act as a directory server.

5. Resource Directory Function Set

   This section defines the REST interfaces between an RD and endpoint
   servers, which is called the Resource Directory Function Set.
   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, and removal interfaces
   defined in this section and MAY support the validation interface.
   For the purpose of validation, an endpoint implementing this
   specification SHOULD support ETag validation on /.well-known/core
   (which is very straightforward for static /.well-known/core link
   documents).

   Resource directory entries are designed to be easily exported to
   other discovery mechanisms such as DNS-SD. For that reason,
   parameters that would meaningfully be mapped to DNS are limited to a
   maximum length of 63 bytes.

5.1. Discovery

   Before an endpoint can make use of an RD, it must first know the RD's
   IP address, port and the path of its RD Function Set. 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 (also see Section 4.1). This section
   defines discovery of the RD using the well-known interface of the
   CoRE Link Format [RFC6690] 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 [RFC6690] with the value "core.rd" in the query string. Likewise, a Resource Type parameter value of "core.rd-lookup" is used to discover the RD Lookup Function Set. 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.

An RD implementation of this specification MUST support query filtering for the rt parameter as defined in [RFC6690].

The discovery request interface is specified as follows:

Interaction: EP -> RD

Method: GET

URI Template: /.well-known/core{?rt}

URI Template Variables:

   rt :=   Resource Type (optional). MAY contain the value
      "core.rd", "core.rd-lookup" or "core.rd*"

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

The following response codes are defined for this interface:

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

Failure: 4.04 "Not Found" 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 endpoint 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, although it is recommended to use default locations where possible.

```
          EP                                              RD
           |                                               |
           | ----- GET /.well-known/core?rt=core.rd* ----->|
           |                                               |
           |                                               |
           | <---- 2.05 Content "</rd>; rt="core.rd" ------ |
           |                                               |
```

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

    Res: 2.05 Content
    </rd>;rt="core.rd",
    </rd-lookup>;rt="core.rd-lookup",
    </rd-group>;rt="core.rd-group"

5.2. Registration

   After discovering the location of an RD Function Set, an endpoint MAY
   register its resources using the registration interface. This
   interface accepts a POST from an endpoint 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 endpoint, its domain and the lifetime of the
   registration. All parameters except the endpoint name are optional.
   It is expected that other specifications MAY define further
   parameters (it is to be determined if a registry of parameters is
   needed for this purpose). The RD then creates a new resource or
   updates an existing resource in the RD and returns its location. An
   endpoint MUST use that location when refreshing registrations using
   this interface. Endpoint resources in the RD are kept active for the
   period indicated by the lifetime parameter. The endpoint is
   responsible for refreshing the entry within this period using either
   the registration or update interface. The registration interface
   MUST be implemented to be idempotent, so that registering twice with
   the same endpoint parameter does not create multiple RD entries.

   The registration request interface is specified as follows:

   Interaction: EP -> RD

   Method: POST

URI Template: /{+rd}{?ep,d,et,lt,con}

URI Template Variables:

   rd :=   RD Function Set path (mandatory). This is the path of the
      RD Function Set. An RD SHOULD use the value "rd" for this
      variable whenever possible.

   ep :=   Endpoint (mandatory). The endpoint identifier or name of
      the registering node, unique within that domain. The maximum
      length of this parameter is 63 bytes.

   d :=   Domain (optional). The domain to which this endpoint
      belongs. The maximum length of this parameter is 63 bytes.
      Optional. When this parameter is elided, the RD MAY associate
      the endpoint with a configured default domain.

   et :=   Endpoint Type (optional). The semantic type of the
      endpoint. The maximum length of this parameter is 63 bytes.
      Optional.

   lt :=   Lifetime (optional). 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.

   con :=   Context (optional). This parameter sets the scheme,
      address and port at which this server is available in the form
      scheme://host:port. Optional. In the absence of this
      parameter the scheme of the protocol, source IP address and
      source port of the register request are assumed.

   Content-Type: application/link-format

   The following response codes are defined for this interface:

   Success: 2.01 "Created". The Location header MUST be included with
      the new resource entry for the endpoint. This Location MUST be a
      stable identifier generated by the RD as it is used for all
      subsequent operations on this registration (update, delete).

   Failure: 4.00 "Bad Request". Malformed request.

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

   The following example shows an endpoint with the name "node1"
   registering two resources to an RD using this interface. The
   resulting location /rd/4521 is just an example of an RD generated

location.


```
      EP                                              RD
       |                                               |
       | --- POST /rd?ep=node1 "</sensors..." ------->  |
       |                                               |
       |                                               |
       | <-- 2.01 Created Location: /rd/4521 ---------- |
       |                                               |
```


   Req: POST coap://rd.example.com/rd?ep=node1
   Payload:
   </sensors/temp>;ct=41;rt="temperature-c";if="sensor",
   </sensors/light>;ct=41;rt="light-lux";if="sensor"

   Res: 2.01 Created
   Location: /rd/4521

5.3. Update

   The update interface is used by an endpoint to refresh or update its
   registration with an RD. To use the interface, the endpoint sends a
   PUT request to the resource returned in the Location option in the
   response to the first registration. An update MAY contain
   registration parameters if there have been changes since the last
   registration or update. Parameters that have not changed SHOULD NOT
   be included in an update. Upon receiving an update request, the RD
   resets the timeout for that endpoint and stores the values of the
   parameters included in the update (if any).

   The update request interface is specified as follows:

   Interaction: EP -> RD

   Method: PUT

   URI Template: /{+location}{?et,lt,con}

   URI Template Variables:

      location :=   This is the Location path returned by the RD as a
         result of a successful registration.

et :=   Endpoint Type (optional). The semantic type of the
         endpoint. The maximum length of this parameter is 63 btyes.
         Optional.

lt :=   Lifetime (optional). 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.

con :=   Context (optional). This parameter sets the scheme,
          address and port at which this server is available in the form
          scheme://host:port. Optional. In the absence of this
          parameter the scheme of the protocol, source IP address and
          source port used to register are assumed.

Content-Type: None

The following response codes are defined for this interface:

Success: 2.04 "Changed" in the update was successfully processed.

Failure: 4.00 "Bad Request". Malformed request.

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

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

```
      EP                                                    RD
       |                                                    |
       | --- PUT /rd/4521  ------------------------->       |
       |                                                    |
       |                                                    |
       | <-- 2.04 Changed  -------------------------        |
       |                                                    |
```

Req: PUT /rd/4521

Res: 2.04 Changed

5.4. Validation

In some cases, an RD may want to validate that it has the latest
version of an endpoint's resources. This can be performed with a GET
on the well-known interface of the CoRE Link Format including the

latest ETag stored for that endpoint. For the purpose of validation,
an endpoint implementing this specification SHOULD support ETag
validation on /.well-known/core.

The validation request interface is specified as follows:

Interaction: RD -> EP

Method: GET

Path: /.well-known/core

Parameters: None

ETag: The ETag option MUST be included

The following responses codes are defined for this interface:

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
   (application/link-format) and its corresponding ETag.

Failure: 4.00 "Bad Request". Malformed request.

The following examples shows a successful validation.

         EP   RD
          | |
          | <--- GET /.well-known/core ETag: 0x40 -------- |
          | |
          | |
          | --- 2.03 Valid -----------------------------> |
          | |
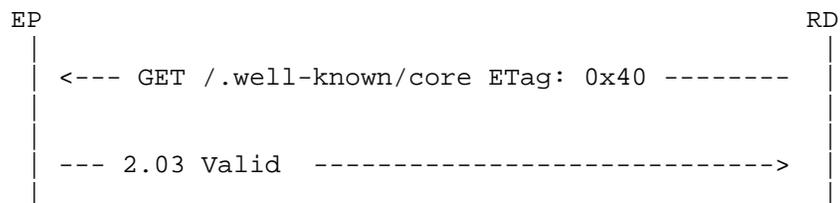
     Req: GET /.well-known/core
     ETag: 0x40

     Res: 2.03 Valid

5.5. Removal

   Although RD entries have soft state and will eventually timeout after
   their lifetime, an endpoint 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 endpoint resource.

   The removal request interface is specified as follows:

   Interaction: EP -> RD

   Method: DELETE

   URI Template: /{+location}

   URI Template Variables:

      location :=   This is the Location path returned by the RD as a
         result of a successful registration.

   The following responses codes are defined for this interface:

   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 endpoint from
   the RD.

      EP   RD
       | |
       | --- DELETE /rd/4521 ------------------------------> |
       | |
       | |
       | <-- 2.02 Deleted -------------------------------- |
       | |
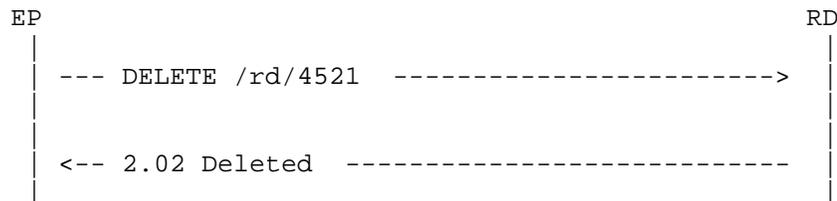
   Req: DELETE /rd/4521

   Res: 2.02 Deleted

6. Group Function Set

   This section defines a function set for the creation of groups of
   endpoints for the purpose of managing and looking up endpoints for
   group operations. The group function set is similar to the resource
   directory function set, in that a group may be created or removed.
   However unlike an endpoint entry, a group entry consists of a list of
   endpoints and does not have a lifetime associated with it. In order
   to make use of multicast requests with CoAP, a group MAY have a
   multicast address associated with it.

6.1. Register a Group

   In order to create a group, a management entity used to configure
   groups, makes a request to the RD indicating the name of the group to
   create (or update), the optional domain the group belongs to, and the
   optional multicast address of the group. The registration message
   includes the list of endpoints that belong to that group. If an
   endpoint has already registered with the RD, the RD attempts to use
   the context of the endpoint from its RD endpoint entry. If the
   client registering the group knows the endpoint has already
   registered, then it MAY send a blank target URI for that endpoint
   link when registering the group.

   The registration request interface is specified as follows:

   Interaction: Manager -> RD

   Method: POST

   URI Template: /{+rd-group}{?gp,d,con}

   URI Template Variables:

      rd-group :=   RD Group Function Set path (mandatory). This is the
         path of the RD Group Function Set. An RD SHOULD use the value
         "rd-group" for this variable whenever possible.

      gp :=   Group Name (mandatory). The name of the group to be
         created or replaced, unique within that domain. The maximum
         length of this parameter is 63 bytes.

      d :=   Domain (optional). The domain to which this group belongs.
         The maximum length of this parameter is 63 bytes. Optional.
         When this parameter is elided, the RD MAY associate the
         endpoint with a configured default domain.

con :=   Context (optional). This parameter is used to set the IP
    multicast address at which this server is available in the form
    scheme://multicast-address:port. Optional. In the absence of
    this parameter no multicast address is configured.

Content-Type: application/link-format

The following response codes are defined for this interface:

Success: 2.01 "Created". The Location header MUST be included with
    the new group entry. This Location MUST be a stable identifier
    generated by the RD as it is used for delete operations on this
    registration.

Failure: 4.00 "Bad Request". Malformed request.

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

The following example shows a group with the name "lights"
registering two endpoints to an RD using this interface. The
resulting location /rd-group/12 is just an example of an RD generated
group location.


```
    EP                                                    RD
     |                                                    |
     | - POST /rd-group?gp=lights "<>;ep=node1..." -->    |
     |                                                    |
     |                                                    |
     | <---- 2.01 Created Location: /rd-group/12 ----     |
     |                                                    |
```

Req: POST coap://rd.example.com/rd-group?gp=lights
Payload:
<>;ep="node1",
<>;ep="node2"

Res: 2.01 Created
Location: /rd-group/12

6.2. Group Removal

A group can be removed simply by sending a removal message to the
location returned when registering the group. Removing a group MUST
NOT remove the endpoints of the group from the RD.

The removal request interface is specified as follows:

Interaction: Manager -> RD

Method: DELETE

URI Template: /{+location}

URI Template Variables:

   location :=   This is the Location path returned by the RD as a
      result of a successful group registration.

The following responses codes are defined for this interface:

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 group from the
RD.

        EP   RD
         | |
         | --- DELETE /rd-group/412 -------------------> |
         | |
         | |
         | <-- 2.02 Deleted --------------------------- |
         | |

   Req: DELETE /rd-group/12

   Res: 2.02 Deleted

7. RD Lookup Function Set

   In order for an RD to be used for discovering resources registered
   with it, a lookup interface can be provided using this function set.
   This lookup interface is defined 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).

This function set allows lookups for domains, groups, endpoints and
resources using attributes defined in the RD Function Set and for use
with the CoRE Link Format. The result of a lookup request is the
list of links (if any) in CoRE Link Format corresponding to the type
of lookup. The target of these links SHOULD be the actual location
of the domain, endpoint or resource, but MAY be an intermediate proxy
e.g. in the case of an HTTP lookup interface for CoAP endpoints.
Multiple query parameters MAY be included in a lookup, all included
parameters MUST match for a resource to be returned. The character
'*' MAY be included at the end of a parameter value as a wildcard
operator.

The lookup interface is specified as follows:

Interaction: Client -> RD

Method: GET

URI Template: /{+rd-lookup-base}/
   {lookup-type}{?d,ep,gp,et,rt,page,count,resource-param}

Parameters:

   rd-lookup-base :=   RD Lookup Function Set path (mandatory). This
      is the path of the RD Lookup Function Set. An RD SHOULD use the
      value "rd-lookup" for this variable whenever possible.

   lookup-type :=   ("d", "ep", "res", "gp") (mandatory) This
      variable is used to select the kind of lookup to perform
      (domain, endpoint or resource).

   ep :=   Endpoint (optional). Used for endpoint, group and
      resource lookups.

   d :=   Domain (optional). Used for domain, group, endpoint and
      resource lookups.

   page :=   Page (optional). Parameter can not be used without the
      count parameter. Results are returned from result set in pages
      that contains 'count' results starting from index (page *
      count).

   count :=   Count (optional). Number of results is limited to this
      parameter value. If the parameter is not present, then an RD
      implementation specific default value SHOULD be used.

rt :=   Resource type (optional). Used for group, endpoint and
    resource lookups.

rt :=   Endpoint type (optional). Used for group, endpoint and
    resource lookups.

resource-param :=   Link attribute parameters (optional). Any
    link attribute as defined in Section 4.1 of [RFC6690], used for
    resource lookups.

The following responses codes are defined for this interface:

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

Failure: 4.04 "Not Found" 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 resource lookup:

    Client   RD
      | |
      | ----- GET /rd-lookup/res?rt=temperature ---------------> |
      | |
      | |
      | <-- 2.05 Content "<coap://node1/temp>;rt="temperature" ---- |
      | |

    Req: GET /rd-lookup/res?rt=temperature

    Res: 2.05 Content
    <coap://{ip:port}/temp>

The following example shows a client performing an endpoint lookup:

```
     Client                                                     RD
        |                                                       |
        | ----- GET /rd-lookup/ep?et=power-node ------------------->  |
        |                                                       |
        |                                                       |
        | <-- 2.05 Content "<coap://{ip:port}>;ep="node5" ----------- |
        |                                                       |
```

```
     Req: GET /rd-lookup/ep?et=power-node

     Res: 2.05 Content
     <coap://{ip:port}>;ep="node5",
     <coap://{ip:port}>;ep="node7"
```

The following example shows a client performing a domain lookup:

```
     Client                                                     RD
        |                                                       |
        | ----- GET /rd-lookup/d ------------------------------->  |
        |                                                       |
        |                                                       |
        | <-- 2.05 Content "</rd>;d=domain1,</rd>;d=domain2 --------- |
        |                                                       |
```

```
     Req: GET /rd-lookup/d

     Res: 2.05 Content
     </rd>;d="domain1",
     </rd>;d="domain2"
```

The following example shows a client performing a group lookup for
all groups:

```
     Client                                                     RD
        |                                                       |
        | ----- GET /rd-lookup/gp ------------------------------>  |
        |                                                       |
        |                                                       |
```

```
          |  <-- 2.05 Content </rd-group/12>;gp="lights1";d="domain1" -- |
          |                                                              |
```

```
     Req: GET /rd-lookup/gp

     Res: 2.05 Content
     </rd-group/12>;gp="lights1";d="domain1"
```

The following example shows a client performing a lookup for all
endpoints in a particular group:

```
     Client                                                           RD
        |                                                             |
        | ----- GET GET /rd-lookup/ep?gp=lights1------------------->  |
        |                                                             |
        |                                                             |
        | <-- 2.05 Content "</rd>;d=domain1,</rd>;d=domain2 --------- |
        |                                                             |
```

```
     Req: GET /rd-lookup/ep?gp=lights1

     Res: 2.05 Content
     <coap://host:port>;ep="node1",
     <coap://host:port>;ep="node2",
```

The following example shows a client performing a lookup for all
groups an endpoint belongs to:

```
     Client                                                           RD
        |                                                             |
        | ----- GET /rd-lookup/gp?ep=node1 ----------------------->   |
        |                                                             |
        |                                                             |
        | <-- 2.05 Content "</rd>;d=domain1,</rd>;d=domain2 --------- |
        |                                                             |
```

Req: GET /rd-lookup/gp?ep=node1

Res: 2.05 Content
<coap://host:port>;gp="lights1";ep="node1",

8. 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
[RFC6690]:

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

8.1. 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 bytes in length. The resource
identifier attribute MUST NOT appear more than once in a link
description.

8.2. 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
endpoints. 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.

9. Security Considerations

   This document needs the same security considerations as described in
   Section 7 of [RFC5988] and Section 6 of [RFC6690]. 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].

   Access control SHOULD be performed separately for the RD Function Set
   and the RD Lookup Function Set, as different endpoints may be
   authorized to register with an RD from those authorized to lookup
   endpoints from the RD. Such access control SHOULD be performed in as
   fine-grained a level as possible. For example access control for
   lookups could be performed either at the domain, endpoint or resource
   level.

10. IANA Considerations

   "core.rd", "core.rd-group" and "core.rd-lookup" resource types need
   to be registered with the resource type registry defined by
   [RFC6690].

   The "exp" attribute needs to be registered when a future Web Linking
   attribute is created.

11. Acknowledgments

   Szymon Sasin, Kerry Lynn, Esko Dijk, Peter van der Stok, Anders
   Brandt, Matthieu Vial, Sampo Ukkola 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.

12. Changelog

   Changes from -04 to -05:

      o Restricted Update to parameter updates.

      o Added pagination support for the Lookup interface.

      o Minor editing, bug fixes and reference updates.

      o Added group support.

o Changed rt= to et= for the registration & update interface

Changes from -03 to -04:

o Added the ins= parameter back for the DNS-SD mapping.

o Integrated the Simple Directory Discovery from Carsten.

o Editorial improvements.

o Fixed the use of ETags.

Changes from -02 to -03:

o Changed the endpoint name back to a single registration
parameter ep= and removed the h= and ins= parameters.

o Updated REST interface descriptions to use RFC6570 URI Template
format.

o Introduced an improved RD Lookup design as its own function set.

o Improved the security considerations section.

o Made the POST registration interface idempotent by requiring the
ep= paramter to be present.

Changes from -01 to -02:

o Added a terminology section.

o Changed the inclusing of an ETag in registration or update to a
MAY.

o Added the concept of an RD Domain and a registration parameter
for it.

o Recommended the Location returned from a registration to be
stable, allowing for endpoint and Domain information to be changed
during updates.

o Changed the lookup interface to accept endpoint and Domain as
query string parameters to control the scope of a lookup.

13. References

13.1. Normative References

   [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.

   [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M.,
              and D. Orchard, "URI Template", RFC 6570, March 2012.

   [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
              Format", RFC 6690, August 2012.

13.2. 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-13 (work in progress), December 2012.

   [I-D.vanderstok-core-bc]
              Stok, P. and K. Lynn, "CoAP Utilization for Building
              Control", draft-vanderstok-core-bc-05 (work in progress),
              October 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.

   [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann,
              "Neighbor Discovery Optimization for IPv6 over Low-Power
              Wireless Personal Area Networks (6LoWPANs)", RFC 6775,
              November 2012.

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

   Carsten Bormann
   Universitaet Bremen TZI
   Postfach 330440
   Bremen D-28359
   Germany

   Phone: +49-421-218-63921
   Fax: +49-421-218-7000
   Email: cabo@tzi.org