CoRE Resource Directory
draft-ietf-core-resource-directory-07

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

Copyright Notice

Table of Contents

1.  Introduction

   The work on Constrained RESTful Environments (CoRE) 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].  However,
   [RFC6690] 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 register,
   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 [RFC7252], they can be applied in an
   equivalent manner to HTTP [RFC7230].

2.  Terminology

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
   "OPTIONAL" in this document are to be interpreted as described in
   [RFC2119].  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
   [RFC7252].  To describe the REST interfaces defined in this
   specification, the URI Template format is used [RFC6570].

   This specification makes use of the following additional terminology:

   Resource Directory
      A 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.  This specification assumes that the list
      of Domains supported by an RD is pre-configured by that RD.  When
      a domain is exported to DNS, the domain value equates to the DNS
      domain name.

   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 are unique.

   Endpoint
      Endpoint (EP) is a term used to describe a web server or client in
      [RFC7252].  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 is unique within
the associated domain of the registration.

3.  Architecture and Use Cases

The resource directory architecture is illustrated 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
scheme, IP address and port (called Context), 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), and for clients to lookup
resources from the RD or maintain groups.  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.  This information
hierachy is shown in Figure 2.

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 an 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.  A lookup
interface for discovering any of the Web Links held in the RD is
provided using the CoRE Link Format.

```
                   Registration    Lookup, Group
                     Interface        Interfaces
     +----+             |               |
     | EP |----         |               |
     +----+    ----     |               |
                  --|-      +------+     |
     +----+         | ----|      |     |   +--------+
     | EP | ---------|-----|  RD  |----|-----| Client |
     +----+         | ----|      |     |   +--------+
                  --|-      +------+     |
     +----+    ----     |               |
     | EP |----         |               |
     +----+
```

Figure 1: The resource directory architecture.

```
              +------------+
              |   Domain   | <-- Name
              +------------+
                 |     |
                 |   +------------+
                 |   |   Group    | <-- Name, IP
                 |   +------------+
                 |      |
              +------------+
              |  Endpoint  | <-- Name, Scheme, IP, Port
              +------------+
                    |
                    |
              +------------+
              |  Resource  | <-- Target, Parameters
              +------------+
```

Figure 2: The resource directory information hierarchy.

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: machines.  The machines are
   connected directly to a mobile network using an 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 fleet management purposes and monitoring
   environment parameters) hosted by the mobile operator or somewhere
   else in the network, periodically a description of its own
   capabilities.  Due to the usual network configuration of mobile
   networks, the EPs attached to the mobile network may not always be
   efficiently reachable.  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 link parameters, obtain information on how to contact them (URLs
of the proxy server) and then initiate interaction to obtain
information that is finally processed, displayed on the screen and
usually stored in a database.  Similarly, fleet management systems
provide the appropriate link parameters 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 discovery requirements of these
   applications are demanding.  Home automation usually relies on run-
   time discovery to commission 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.

3.3.  Use Case: Link Catalogues

   Resources may be shared through data brokers that have no knowledge
   beforehand of who is going to consume the data.  Resource Directory
   can be used to hold links about resources and services hosted
   anywhere to make them discoverable by a general class of
   applications.

   For example, environmental and weather sensors that generate data for
   public consumption may provide the data to an intermediary server, or
   broker.  Sensor data are published to the intermediary upon changes
   or at regular intervals.  Descriptions of the sensors that resolve to
   links to sensor data may be published to a Resource Directory.
   Applications wishing to consume the data can use the Resource
   Directory lookup function set to discover and resolve links to the
   desired resources and endpoints.  The Resource Directory service need
   not be coupled with the data intermediary service.  Mapping of
   Resource Directories to data intermediaries may be many-to-many.

   Metadata in web link compatible representations are supplied by
   Resource Directories, which may be internally stored as triples, or
   relation/attribute pairs providing metadata about resource links.

External catalogs that are represented in other formats may be
converted to common web linking formats for storage and access by
Resource Directories.  Since it is common practice for these to be
URN encoded, simple and lossless structural transforms should
generally be sufficient to store external metadata in Resource
Directories.

The additional features of Resource Directory allow domains to be
defined to enable access to a particular set of resources from
particular applications.  This provides isolation and protection of
sensitive data when needed.  Resource groups may defined to allow
batched reads from multiple resources.

4.  Simple Directory Discovery

Not all endpoints hosting resources are expected to know how to
implement the Resource Directory Function Set (see Section 5) 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 available the hosted
resources that it wants to be discovered, 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 non-empty 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 GET requests to that well-known multicast
   address (see Section 5.1).

   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.  For
   example, 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.

4.2.  Third-party registration

   For some applications, even Simple Directory Discovery may be too
   taxing for certain very constrained devices, in particular if the
   security requirements become too onerous.

   In a controlled environment (e.g. building control), the Resource
   Directory can be filled by a third device, called an installation
   tool.  The installation tool can fill the Resource Directory from a
   database or other means.  For that purpose the scheme, IP address and
   port of the registered device is indicated in the Context parameter
   of the registration as well.

5.  Resource Directory Function Set

   This section defines the REST interfaces between an RD and endpoints,
   which is called the Resource Directory Function Set. Although the
   examples throughout this section assume the use of CoAP [RFC7252],
   these REST interfaces can also be realized using HTTP [RFC7230].  In
   all definitions in this section, both CoAP response codes (with dot
   notation) and HTTP response codes (without dot notation) are shown.
   An RD implementing this specification MUST support the discovery,
   registration, update, lookup, and removal interfaces defined in this
   section.

   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 SHOULD be 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 (see also 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.

A Resource Directory MAY provide hints about the content-formats it
supports in the links it exposes or registers, using the "ct" link
attribute, as shown in the example below.  Clients MAY use these
hints to select alternate content-formats for interaction with the
Resource Directory.

HTTP does not support multicast and consequently only unicast
discovery can be supported using HTTP.  Links to Resource Directories
MAY be registered in other Resource Directories, and well-known entry
points SHOULD be provided to enable the bootstrapping of unicast
discovery.

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 one or more of the
      values "core.rd", "core.rd-lookup", "core.rd-group" or
      "core.rd*"

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

Content-Format:  application/link-format+json (if any)

Content-Format:  application/link-format+cbor (if any)

The following response codes are defined for this interface:

Success:  2.05 "Content" with an application/link-format,
   application/link-format+json, or application/link-format+cbor
   payload containing one or more matching entries for the RD
   resource.

   Failure:  4.04 "Not Found" is returned in case no matching entry is
      found for a unicast request.

   Failure:  4.00 "Bad Request" is returned in case of a malformed
      request for a unicast request.

   Failure:  No error response to a multicast request.

   HTTP support :  NO

   The following example shows an endpoint discovering an RD using this
   interface, thus learning that the base RD resource is, in this
   example, at /rd.  Note that it is up to the RD to choose its base RD
   resource, although diagnostics and debugging is facilitated by using
   the base paths specified here 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";ct="application/link-format+cbor",
   </rd-lookup>;rt="core.rd-lookup";ct="application/link-format+cbor",
   </rd-group>;rt="core.rd-group";ct="application/link-format+cbor"

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 [RFC6690], JSON CoRE Link Format (application/link-
   format+json), or CBOR CoRE Link Format (application/link-format+cbor)
   [I-D.ietf-core-links-json], 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 will define
   further parameters (see Section 11.3).  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, as obtained from discovery.  An RD SHOULD use
      the value "rd" for this variable whenever possible.

   ep :=  Endpoint name (mandatory).  The endpoint name is an
      identifier that MUST be unique within a domain.  The maximum
      length of this parameter is 63 bytes.

   d :=  Domain (optional).  The domain to which this endpoint
      belongs.  This parameter SHOULD be less than 63 bytes.
      Optional.  When this parameter is elided, the RD MAY associate
      the endpoint with a configured default domain.  The domain
      value is needed to export the endpoint to DNS-SD (see
      Section 9).

   et :=  Endpoint Type (optional).  The semantic type of the
      endpoint.  This parameter SHOULD be less than 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.  This
      parameter is mandatory when the directory is filled by a third
      party such as an installation tool.

Content-Format:  application/link-format

Content-Format:  application/link-format+json

Content-Format:  application/link-format+cbor

The following response codes are defined for this interface:

Success:  2.01 "Created" or 201 "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.  The
   resource returned in the Location is only for the purpose of the
   Update (POST) and Removal (DELETE), and MUST NOT implement GET or
   PUT methods.

Failure:  4.00 "Bad Request" or 400 "Bad Request".  Malformed
   request.

Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
   Service could not perform the operation.

HTTP support:  YES

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
Content-Format: 40
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

```
Req: POST /rd?ep=node1 HTTP/1.1
Host : example.com
Content-Format: application/link-format
Payload:
</sensors/temp>;ct=41;rt="temperature-c";if="sensor",
</sensors/light>;ct=41;rt="light-lux";if="sensor"

Res: 201 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
POST request to the resource returned in the Location option in the
response to the first registration.  An update MAY update the
lifetime or context parameters if they have changed 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 updates the scheme, IP
address and port of the endpoint (using the source address of the
update, or the context parameter if present).

An update MAY optionally add or replace links for the endpoint by
including those links in the payload of the update as a CoRE Link
Format document.  Including links in an update message greatly
increases the load on an RD and SHOULD be done infrequently.  A link
is replaced only if both the target URI and relation type match (see
Section 10.1)

The update request interface is specified as follows:

Interaction:  EP -> RD

Method:  POST

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

URI Template Variables:

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

   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.  This parameter is
   compulsory when the directory is filled by a third party such
   as an installation tool.

Content-Format:  application/link-format (optional)

Content-Format:  application/link-format+json (optional)

Content-Format:  application/link-format+cbor (optional)

The following response codes are defined for this interface:

Success:  2.04 "Changed" or 204 "No Content" if the update was
   successfully processed.

Failure:  4.00 "Bad Request" or 400 "Bad Request".  Malformed
   request.

Failure:  4.04 "Not Found" or 404 "Not Found".  Registration does not
   exist (e.g. may have expired).

Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
   Service could not perform the operation.

HTTP support:  YES

The following example shows an endpoint updating its registration at
an RD using this interface.

```
    EP                                                  RD
     |                                                   |
     | --- POST /rd/4521  ------------------------->     |
     |                                                   |
     |                                                   |
     | <-- 2.04 Changed  --------------------------      |
     |                                                   |
```
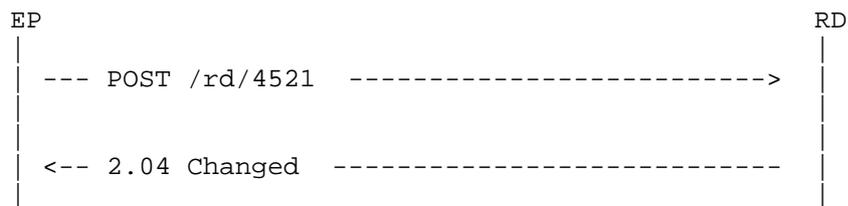
Req: POST /rd/4521

Res: 2.04 Changed

5.4.  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 earlier registration.

   The following responses codes are defined for this interface:

   Success:  2.02 "Deleted" or 204 "No Content" upon successful deletion

   Failure:  4.00 "Bad Request" or 400 "Bad request".  Malformed
      request.

   Failure:  4.04 "Not Found" or 404 "Not Found".  Registration does not
      exist (e.g. may have expired).

   Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
      Service could not perform the operation.

   HTTP support: YES

   The following examples shows successful removal of the endpoint from
   the RD.

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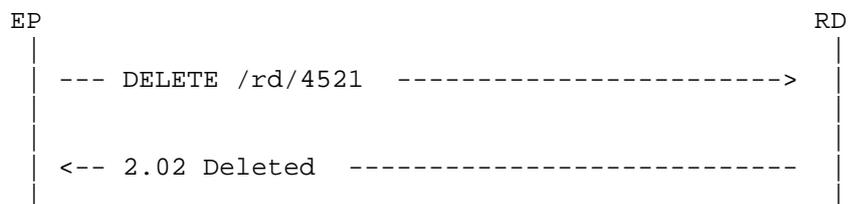
     Req: DELETE /rd/4521

     Res: 2.02 Deleted

5.5.  Read Endpoint Links

     Some endpoints may wish to manage their links as a collection, and
     may need to read the current set of links in order to determine link
     maintenance operations.

     One or more links MAY be selected by using query filtering as
     specified in [RFC6690] Section 4.1

     The read request interface is specified as follows:

     Interaction:  EP -> RD

     Method:  GET

     URI Template:  /{+location}{?href,rel,rt,if,ct}

     URI Template Variables:

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

        href,rel,rt,if,ct := link relations and attributes specified in
        the query in order to select particular links based on their
        relations and attributes. "href" denotes the URI target of the
        link.  See [RFC6690] Sec. 4.1

     The following responses codes are defined for this interface:

     Success:  2.05 "Content" or 200 "OK" upon success with an
        "application/link-format", "application/link-format+cbor", or
        "application/link-format+json" payload.

     Failure:  4.00 "Bad Request" or 400 "Bad Request".  Malformed
        request.

     Failure:  4.04 "Not Found" or 404 "Not Found".  Registration does not
        exist (e.g. may have expired).

     Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
        Service could not perform the operation.

     HTTP support: YES

The following examples show successful read of the endpoint links
from the RD.

```
     EP                                                        RD
      |                                                         |
      | --- GET /rd/4521  ------------------------>  |
      |                                                         |
      |                                                         |
      | <-- 2.05 Content </sensors... ---------------  |
      |                                                         |
```
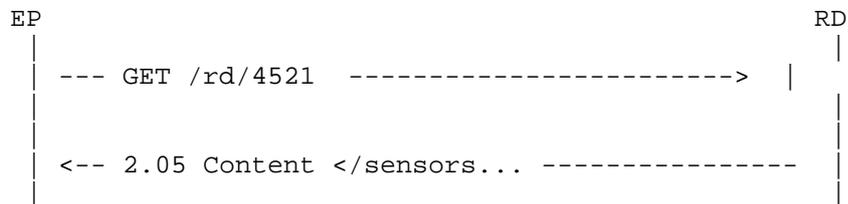
Req: GET /rd/4521

Res: 2.01 Content
Payload:
</sensors/temp>;ct=41;rt="temperature-c";if="sensor",
</sensors/light>;ct=41;rt="light-lux";if="sensor"

5.6.  Update Endpoint Links

[This section will be removed before or as a result of a working-
group last-call if the PATCH methods do not achieve the same level of
consensus as the present document.]

A PATCH update adds, removes or changes links for the endpoint by
including link update information in the payload of the update as a
merge-patch+json format [RFC7396] document.

One or more links are selected for update by using query filtering as
specified in [RFC6690] Section 4.1

The query filter selects the links to be modified or deleted, by
matching the query parameter values to the values of the link
attributes.

When the query parameters are not present in the request, the payload
specifies links to be added to the target document.  When the query
parameters are present, the attribute names and values in the query
parameters select one or more links on which to apply the PATCH
operation.

If an attribute name specified in the PATCH document exists in any
the set of selected links, all occurrences of the attribute value in
the target document MUST be updated using the value from the PATCH
payload.  If the attribute name is not present in any selected links,
the attribute MUST be added to the links.

The update request interface is specified as follows:

Interaction:  EP -> RD

Method:  PATCH

URI Template:  /{+location}{?href,rel,rt,if,ct}

URI Template Variables:

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

   href,rel,rt,if,ct := link relations and attributes specified in
   the query in order to select particular links based on their
   relations and attributes. "href" denotes the URI target of the
   link.  See [RFC6690] Sec. 4.1

Content-Format:  application/merge-patch+json (mandatory)

The following response codes are defined for this interface:

Success:  2.04 "Changed" 0r 204 "No Content" in the update was
   successfully processed.

Failure:  4.00 "Bad Request" or 400 "Bad Request".  Malformed
   request.

Failure:  4.04 "Not Found" or 404 "Not Found".  Registration resource
   does not exist (e.g. may have expired).

Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
   Service could not perform the operation.

HTTP support: YES

The following examples show an endpoint adding </sensors/humid>,
modifying </sensors/temp>, and removing </sensors/light> links in RD
using the Update Endpoint Links function.

The following example shows an EP adding the link </sensors/
humid>;ct=41;rt="humidity-s";if="sensor" to the collection of links
at the location /rd/4521.

```
        EP                                               RD
         |                                                |
         | --- PATCH /rd/4521---------------------------> |
         |                                                |
         |                                                |
         | <-- 2.04 Changed  ---------------------------  |
         |                                                |
```
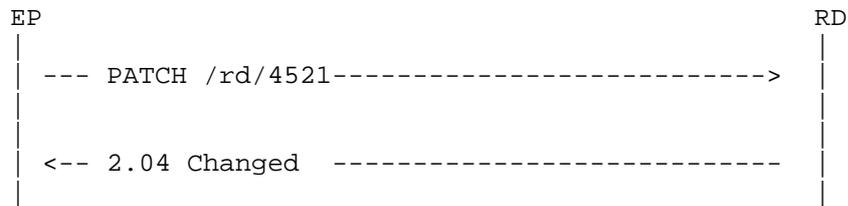
Req: PATCH /rd/4521

Payload:
[{"href":"/sensors/humid","ct": 41, "rt": "humidity-s", "if": "sensor"}]

Content-Format:
application/merge-patch+json

Res: 2.04 Changed

   The following example shows an EP modifying all links at the location
   /rd/4521 which are identified by href="/sensors/temp", from the
   initial link-value of </sensors/temp>;rt="temperature" to the new
   link-value </sensors/temp>;rt="temperature-c";if="sensor" by changing
   the value of the link attribute "rt" and adding the link attribute
   if="sensor" using the PATCH operation with the supplied merge-
   patch+json document payload.

```
        EP                                               RD
         |                                                |
         | --- PATCH /rd/4521?href="/sensors/temp"  ----> |
         |                                                |
         |                                                |
         | <-- 2.04 Changed  ---------------------------  |
         |                                                |
```
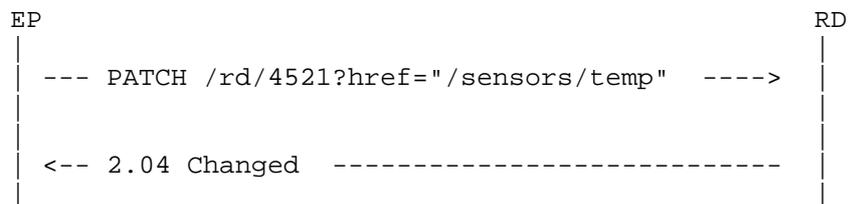
   Req: PATCH /rd/4521?href="/sensors/temp"

   Payload:
   {"rt": "temperature-c", "if": "sensor"},

   Content-Format:
   application/merge-patch+json

   Res: 2.04 Changed

   This example shows an EP removing all links at the location /rd/4521
   which are identified by href="/sensors/light".

```
      EP                                                    RD
       |                                                     |
       | --- PATCH /rd/4521?href="/sensors/light"  ---->     |
       |                                                     |
       |                                                     |
       | <-- 2.04 Changed  --------------------------        |
       |                                                     |
```
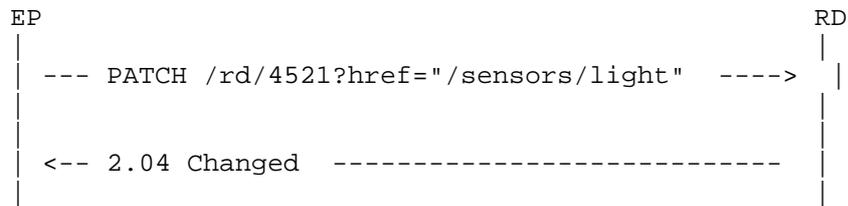
    Req: PATCH /rd/4521?href="/sensors/light"

    Payload:
    {null}

    Content-Format:
    application/merge-patch+json

    Res: 2.04 Changed

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), optionally the domain the group belongs to, and
    optionally the 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.  Configuration of the endpoints
    themselves is out of scope of this specification.  Such an interface
    for managing the group membership of an endpoint has been defined in
    [RFC7390].

    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.  The domain value is
      needed to export the endpoint to DNS-SD (see Section 9)

   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.  This
      parameter is compulsory when the directory is filled by an
      installation tool.

Content-Format:  application/link-format

Content-Format:  application/link-format+json

Content-Format:  application/link-format+cbor

The following response codes are defined for this interface:

Success:  2.01 "Created" or 201 "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" or 400 "Bad Request".  Malformed
   request.

Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
   Service could not perform the operation.

HTTP support:  YES

The following example shows an EP registering a group with the name
"lights" which has 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

Req: POST /rd-group?gp=lights HTTP/1.1
Host: example.com
Accept: application/link-format
Payload:
<>;ep="node1",
<>;ep="node2"

Res: 201 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" or 204 "No Content" upon successful deletion

   Failure:  4.00 "Bad Request" or 400 "Bad Request".  Malformed
      request.

   Failure:  4.04 "Not Found" or 404 "Not Found".  Group does not exist.

   Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
      Service could not perform the operation.

   HTTP support:  YES

   The following examples shows successful removal of the group from the
   RD.

```
      EP                                                      RD
       |                                                       |
       | --- DELETE /rd-group/412  ------------------->        |
       |                                                       |
       |                                                       |
       | <-- 2.02 Deleted  ---------------------------         |
       |                                                       |
```
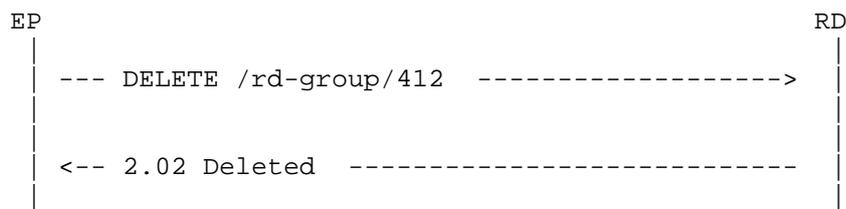
   Req: DELETE /rd-group/12

   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) corresponding to the type of lookup.  Using
   the Accept Option, the requester can control whether this list is
   returned in CoRE Link Format ("application/link-format", default) or

   its alternate content-formats ("application/link-format+json" or
   "application/link-format+cbor").
   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}

   URI Template Variables:

      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.

   Content-Format:  application/link-format (optional)

   Content-Format:  application/link-format+json (optional)

   Content-Format:  application/link-format+cbor (optional)

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

      ep :=  Endpoint name (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.

et :=  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" or 200 "OK" with an "application/link-
    format", "application/link-format+cbor", or "application/link-
    format+json" payload containing matching entries for the lookup.

Failure:  4.04 "Not Found" or 404 "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" or 400 "Bad Request".  Malformed
    request.

Failure:  5.03 "Service Unavailable" or 503 "Service Unavailable".
    Service could not perform the operation.

HTTP support:  YES

The examples in this section assume a CoAP host with IP address
FDFD::123 and a default CoAP port 61616.  HTTP hosts are possible and
do not change the nature of the examples.  The following example
shows a client performing a resource lookup:

```
    Client                                                       RD
       |                                                          |
       | ----- GET /rd-lookup/res?rt=temperature ---------------> |
       |                                                          |
       |                                                          |
       | <-- 2.05 Content <coap://[FDFD::123]:61616/temp>;--------- |
       |                                   rt="temperature" -------- |
       |                                                          |
```
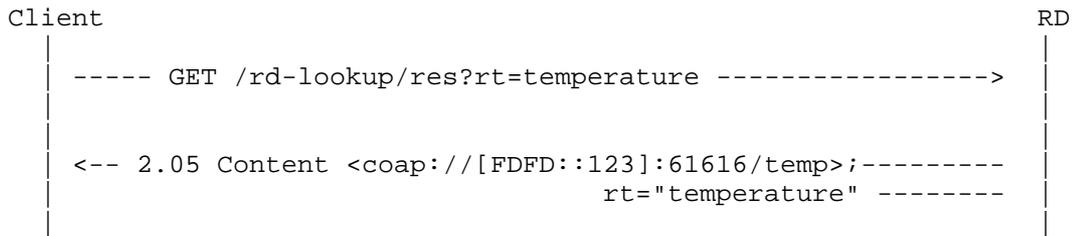
Req: GET /rd-lookup/res?rt=temperature

Res: 2.05 Content
<coap://[FDFD::123]:61616/temp>;rt="temperature"

The following example shows a client performing an endpoint type
lookup:

```
    Client                                                      RD
       |                                                         |
       | ----- GET /rd-lookup/ep?et=power-node -------------------> |
       |                                                         |
       |                                                         |
       | <-- 2.05 Content <coap://[FDFD::123]:61616>;ep="node5" ---- |
       |                                                         |
```

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

Res: 2.05 Content
<coap://[FDFD::123]:61616>;ep="node5",
<coap://[FDFD::123]:61616>;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="example.com" ------------- |
       |                                                         |
```

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

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

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

```
      Client                                                     RD
        |                                                         |
        | ----- GET /rd-lookup/ep?gp=lights1--------------------> |
        |                                                         |
        |                                                         |
        | <-- 2.05 Content <coap://[FDFD::123]:61616>;ep="node1" ---- |
        |                                                         |
```

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

   Res: 2.05 Content
   <coap://[FDFD::123]:61616>;ep="node1",
   <coap://[FDFD::123]:61616>;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 <coap://[FDFD::123]:61616>;gp="lights1"; -- |
        |                                         ep="node1" ------ |
        |                                                         |
```

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

   Res: 2.05 Content
   <coap://[FDFD::123]:61616>;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 attribute 'ins'

   The Resource Instance "ins" attribute is an identifier for this
   resource, which makes it possible to distinguish it from other
   similar resources.  This attribute is similar in use to the
   <Instance> portion of a DNS-SD record (see Section 9.1, 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 the directory.

   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 attribute 'exp'

   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.  DNS-SD Mapping

   CoRE Resource Discovery is intended to support fine-grained discovery
   of hosted resources, their attributes, and possibly other resource
   relations [RFC6690].  In contrast, service discovery generally refers
   to a coarse-grained resolution of an endpoint's IP address, port
   number, and protocol.

   Resource and service discovery are complementary in the case of large
   networks, where the latter can facilitate scaling.  This document
   defines a mapping between CoRE Link Format attributes and DNS-Based
   Service Discovery [RFC6763] fields that permits discovery of CoAP
   services by either method.

9.1.  DNS-based Service discovery

   DNS-Based Service Discovery (DNS-SD) defines a conventional method of
   configuring DNS PTR, SRV, and TXT resource records to facilitate
   discovery of services (such as CoAP servers in a subdomain) using the
   existing DNS infrastructure.  This section gives a brief overview of
   DNS-SD; see [RFC6763] for a detailed specification.

   DNS-SD service names are limited to 255 octets and are of the form:

   Service Name = <Instance>.<ServiceType>.<Domain>.

   The service name is the label of SRV/TXT resource records.  The SRV
   RR specifies the host and the port of the endpoint.  The TXT RR
   provides additional information in the form of key/value pairs.

   The <Domain> part of the service name is identical to the global (DNS
   subdomain) part of the authority in URIs that identify servers or
   groups of servers.

   The <ServiceType> part is composed of at least two labels.  The first
   label of the pair is the application protocol name [RFC6335] preceded
   by an underscore character.  The second label indicates the transport
   and is always "_udp" for UDP-based CoAP services.  In cases where
   narrowing the scope of the search may be useful, these labels may be
   optionally preceded by a subtype name followed by the "_sub" label.
   An example of this more specific <ServiceType> is
   "light._sub._dali._udp".

   A default <Instance> part of the service name may be set at the
   factory or during the commissioning process.  It SHOULD uniquely
   identify an instance of <ServiceType> within a <Domain>.  Taken
   together, these three elements comprise a unique name for an SRV/ TXT
   record pair within the DNS subdomain.

   The granularity of a service name MAY be that of a host or group, or
   it could represent a particular resource within a CoAP server.  The
   SRV record contains the host name (AAAA record name) and port of the
   service while protocol is part of the service name.  In the case
   where a service name identifies a particular resource, the path part
   of the URI must be carried in a corresponding TXT record.

   A DNS TXT record is in practice limited to a few hundred octets in
   length, which is indicated in the resource record header in the DNS
   response message.  The data consists of one or more strings
   comprising a key=value pair.  By convention, the first pair is
   txtver=<number> (to support different versions of a service
   description).

9.2.  mapping ins to <Instance>

   The Resource Instance "ins" attribute maps to the <Instance> part of
   a DNS-SD service name.  It is stored directly in the DNS as a single
   DNS label of canonical precomposed UTF-8 [RFC3629] "Net-Unicode"
   (Unicode Normalization Form C) [RFC5198] text.  However, to the
   extent that the "ins" attribute may be chosen to match the DNS host
   name of a service, it SHOULD use the syntax defined in Section 3.5 of
   [RFC1034] and Section 2.1 of [RFC1123].

   The <Instance> part of the name of a service being offered on the
   network SHOULD be configurable by the user setting up the service, so
   that he or she may give it an informative name.  However, the device
   or service SHOULD NOT require the user to configure a name before it
   can be used.  A sensible choice of default name can allow the device
   or service to be accessed in many cases without any manual
   configuration at all.  The default name should be short and
   descriptive, and MAY include a collision-resistant substring such as
   the lower bits of the device's MAC address, serial number,
   fingerprint, or other identifier in an attempt to make the name
   relatively unique.

   DNS labels are currently limited to 63 octets in length and the
   entire service name may not exceed 255 octets.

9.3.  Mapping rt to <ServiceType>

   The resource type "rt" attribute is mapped into the <ServiceType>
   part of a DNS-SD service name and SHOULD conform to the reg-rel-type
   production of the Link Format defined in Section 2 of [RFC6690].  The
   "rt" attribute MUST be composed of at least a single Net-Unicode text
   string, without underscore '_' or period '.' and limited to 15 octets
   in length, which represents the application protocol name.  This
   string is mapped to the DNS-SD <ServiceType> by prepending an
   underscore and appending a period followed by the "_udp" label.  For
   example, rt="dali" is mapped into "_dali._udp".

   The application protocol name may be optionally followed by a period
   and a service subtype name consisting of a Net-Unicode text string,
   without underscore or period and limited to 63 octets.  This string
   is mapped to the DNS-SD <ServiceType> by appending a period followed
   by the "_sub" label and then appending a period followed by the
   service type label pair derived as in the previous paragraph.  For
   example, rt="dali.light" is mapped into "light._sub._dali._udp".

   The resulting string is used to form labels for DNS-SD records which
   are stored directly in the DNS.

9.4.  Domain mapping

   DNS domains may be derived from the "d" attribute.  The domain
   attribute may be suffixed with the zone name of the authoritative DNS
   server to generate the domain name.  The "ep" attribute is prefixed
   to the domain name to generate the FQDN to be stored into DNS with an
   AAAA RR.

9.5.  TXT Record key=value strings

   A number of [RFC6763] key/value pairs are derived from link-format
   information, to be exported in the DNS-SD as key=value strings in a
   TXT record ([RFC6763], Section 6.3).

   The resource <URI> is exported as key/value pair "path=<URI>".

   The Interface Description "if" attribute is exported as key/value
   pair "if=<Interface Description>".

   The DNS TXT record can be further populated by importing any other
   resource description attributes as they share the same key=value
   format specified in Section 6 of [RFC6763].

9.6.  Importing resource links into DNS-SD

   Assuming the ability to query a Resource Directory or multicast a GET
   (?exp) over the local link, CoAP resource discovery may be used to
   populate the DNS-SD database in an automated fashion.  CoAP resource
   descriptions (links) can be exported to DNS-SD for exposure to
   service discovery by using the Resource Instance attribute as the
   basis for a unique service name, composed with the Resource Type as
   the <ServiceType>, and registered in the correct <Domain>.  The agent
   responsible for exporting records to the DNS zone file SHOULD be
   authenticated to the DNS server.  The following example shows an
   agent discovering a resource to be exported:

```
     Agent                                                         RD
       |                                                            |
       | --- GET /rd-lookup/res?exp ------------------------------> |
       |                                                            |
       |                                                            |
       | <-- 2.05 Content "<coap://[FDFD::1234]:5683/light/1>;exp;  |
       |                    rt="dali.light";ins="Spot";             |
       |                    d="office";ep="node1"                   |
       |                                                            |
```

```
Req: GET /rd-lookup/res?exp

Res: 2.05 Content
<coap://[FDFD::1234]:5683/light/1>;
  exp;rt="dali.light";ins="Spot";
          d="office";ep="node1"
```

The agent subsequently registers the following DNS-SD RRs, assuming a
zone name "example.com" prefixed with "office":

```
node1.office.example.com.          IN AAAA         FDFD::1234
_dali._udp.office.example.com      IN PTR
                          Spot._dali._udp.office.example.com
light._sub._dali._udp.example.com  IN PTR
                          Spot._dali._udp.office.example.com
Spot._dali._udp.office.example.com IN SRV  0 0 5683
                          node1.office.example.com.
Spot._dali._udp.office.example.com IN TXT
                          txtver=1;path=/light/1
```

In the above figure the Service Name is chosen as
Spot._dali._udp.office.example.com without the light._sub service
prefix.  An alternative Service Name would be:
Spot.light._sub._dali._udp.office.example.com.

10.  Security Considerations

   The security considerations as described in Section 7 of [RFC5988]
   and Section 6 of [RFC6690] apply.  The "/.well-known/core" resource
   may be protected e.g. using DTLS when hosted on a CoAP server as
   described in [RFC7252].  DTLS or TLS based security SHOULD be used on
   all resource directory interfaces defined in this document.

10.1.  Endpoint Identification and Authentication

   An Endpoint is determined to be unique by an RD by the Endpoint
   identifier parameter included during Registration, and any associated
   TLS or DTLS security bindings.  An Endpoint MUST NOT be identified by
   its protocol, port or IP address as these may change over the
   lifetime of an Endpoint.

   Every operation performed by an Endpoint or Client on a resource
   directory SHOULD be mutually authenticated using Pre-Shared Key, Raw
   Public Key or Certificate based security.  Endpoints using a
   Certificate MUST include the Endpoint identifier as the Subject of
   the Certificate, and this identifier MUST be checked by a resource

directory to match the Endpoint identifier included in the
Registration message.

## 10.2.  Access Control

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.

## 10.3.  Denial of Service Attacks

Services that run over UDP unprotected are vulnerable to unknowingly
become part of a DDoS attack as UDP does not require return
routability check.  Therefore, an attacker can easily spoof the
source IP of the target entity and send requests to such a service
which would then respond to the target entity.  This can be used for
large-scale DDoS attacks on the target.  Especially, if the service
returns a response that is order of magnitudes larger than the
request, the situation becomes even worse as now the attack can be
amplified.  DNS servers have been widely used for DDoS amplification
attacks.  Recently, it has been observed that NTP Servers, that also
run on unprotected UDP have been used for DDoS attacks
(http://tools.cisco.com/security/center/content/CiscoSecurityNotice/
CVE-2013-5211) since there is no return routability check and can
have a large amplification factor.  The responses from the NTP server
were found to be 19 times larger than the request.  A Resource
Directory (RD) which responds to wild-card lookups is potentially
vulnerable if run with CoAP over UDP.  Since there is no return
routability check and the responses can be significantly larger than
requests, RDs can unknowingly become part of a DDoS amplification
attack.  Therefore, it is RECOMMENDED that implementations ensure
return routability.  This can be done, for example by responding to
wild card lookups only over DTLS or TLS or TCP.

## 11.  IANA Considerations

## 11.1.  Resource Types

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

11.2.  Link Extension

   The "exp" attribute needs to be registered when a future Web Linking
   link-extension registry is created (e.g. in RFC5988bis).

11.3.  RD Parameter Registry

   This specification defines a new sub-registry for registration and
   lookup parameters called "RD Parameters" under "CoRE Parameters".
   Although this specification defines a basic set of parameters, it is
   expected that other standards that make use of this interface will
   define new ones.

   Each entry in the registry must include the human readable name of
   the parameter, the query parameter, validity requirements if any and
   a description.  The query parameter MUST be a valid URI query key
   [RFC3986].

   Initial entries in this sub-registry are as follows:

   +----------+-------+--------------+------------------------------+
   | Name     | Query | Validity     | Description                  |
   +----------+-------+--------------+------------------------------+
   | Endpoint | ep    |              | Name of the endpoint         |
   | Name     |       |              |                              |
   | Lifetime | lt    | 60-4294967295| Lifetime of the registration |
   |          |       |              | in seconds                   |
   | Domain   | d     |              | Domain to which this endpoint|
   |          |       |              | belongs                      |
   | Endpoint | et    |              | Semantic name of the endpoint|
   | Type     |       |              |                              |
   | Context  | con   | URI          | The scheme, address and port |
   |          |       |              | at which this server is      |
   |          |       |              | available                    |
   | Endpoint | ep    |              | Name of the endpoint, max 63 |
   | Name     |       |              | bytes                        |
   | Group    | gp    |              | Name of a group in the RD    |
   | Name     |       |              |                              |
   | Page     | page  | Integer      | Used for pagination          |
   | Count    | count | Integer      | Used for pagination          |
   +----------+-------+--------------+------------------------------+

                        Table 1: RD Parameters

   The IANA policy for future additions to the sub-registry is "Expert
   Review" as described in [RFC5226].

12.  Examples

   Examples are added here.

12.1.  Lighting Installation

   This example shows a simplified lighting installation which makes use
   of the Resource Directory (RD) with a CoAP interface to facilitate
   the installation and start up of the application code in the lights
   and sensors.  In particular, the example leads to the definition of a
   group and the enabling of the corresponding multicast address.  No
   conclusions must be drawn on the realization of actual installation
   procedures, because the example "emphasizes" some of the issues that
   may influence the use of the RD.

12.1.1.  Installation Characteristics

   The example assumes that the installation is managed.  That means
   that a Commissioning Tool (CT) is used to authorize the addition of
   nodes, name them, and name their services.  The CT can be connected
   to the installation in many ways: the CT can be part of the
   installation network, connected by WiFi to the installation network,
   or connected via GPRS link, or other method.

   It is assumed that there are two naming authorities for the
   installation: (1) the network manager that is responsible for the
   correct operation of the network and the connected interfaces, and
   (2) the lighting manager that is responsible for the correct
   functioning of networked lights and sensors.  The result is the
   existence of two naming schemes coming from the two managing
   entities.

   The example installation consists of one presence sensor, and two
   luminaries, luminary1 and luminary2, each with their own wireless
   interface.  Each luminary contains three lamps: left, right and
   middle.  Each luminary is accessible through one end-point.  For each
   lamp a resource exists to modify the settings of a lamp in a
   luminary.  The purpose of the installation is that the presence
   sensor notifies the presence of persons to a group of lamps.  The
   group of lamps consists of: middle and left lamps of luminary1 and
   right lamp of luminary2.

   Before commissioning by the lighting manager, the network is
   installed and access to the interfaces is proven to work by the
   network manager.  Following the lay-out of cables and routers the
   network manager has defined DNS domains.  The presence sensor and
   luminary1 are part of DNS domain: rtr_5612_rrt.example.com and
   luminary2 is part of rtr_7899_pfa.example.com.  The names of

luminary1- luminary2-, and sensor- interfaces are respectively:
lm_12-345-678, lm_12-456-378, and sn_12-345-781.  These names are
stored in DNS together with their IP addresses.  The FQDN of the
interfaces is shown in Table 2 below:

```
+--------------------+---------------------------------------+
| Name               | FQDN                                  |
+--------------------+---------------------------------------+
| luminary1          | lm_12-345-678.rtr_5612_rrt.example.com |
| luminary2          | lm_12-456-378.rtr_7899_pfa.example.com |
| Presence sensor    | sn_12-345-781.rtr_5612_rrt.example.com |
| Resource directory | pc_123456.rtr_5612_rrt.example.com    |
+--------------------+---------------------------------------+
```

Table 2: interface FQDNs

At the moment of installation, the network under installation is not
necessarily connected to the DNS infra structure.  Therefore, SLAAC
IPv6 addresses are assigned to CT, RD, luminaries and sensor shown in
Table 3 below:

```
+--------------------+--------------+
| Name               | IPv6 address |
+--------------------+--------------+
| luminary1          | FDFD::ABCD:1 |
| luminary2          | FDFD::ABCD:2 |
| Presence sensor    | FDFD::ABCD:3 |
| Resource directory | FDFD::ABCD:0 |
+--------------------+--------------+
```

Table 3: interface SLAAC addresses

In Section 12.1.2 the use of resource directory during installation
is presented.  In Section 12.1.3 the connection to DNS is discussed.

12.1.2.  RD entries

It is assumed that access to the DNS infrastructure is not always
possible during installation.  Therefore, the SLAAC addresses are
used in this section.

For discovery, the resource types (rt) of the devices are important.
The lamps in the luminaries have rt: light, and the presence sensor
has rt: p-sensor.  The end-points have names which are relevant to
the light installation manager.  In this case luminary1, luminary2,
and the presence sensor are located in room 2-4-015, where luminary1
is located at the window and luminary2 and the presence sensor are
located at the door.  The end-point names reflect this physical

location.  The middle, left and right lamps are accessed via path
/light/middle, /light/left, and /light/right respectively.  The
identifiers relevant to the Resource Directory are shown in Table 4
below:

```
+----------------+-----------------+---------------+---------------+
| Name           | end-point       | resource path | resource type |
+----------------+-----------------+---------------+---------------+
| luminary1      | lm_R2-4-015_wndw | /light/left  | light         |
| luminary1      | lm_R2-4-015_wndw | /light/middle| light         |
| luminary1      | lm_R2-4-015_wndw | /light/right | light         |
| luminary2      | lm_R2-4-015_door | /light/left  | light         |
| luminary2      | lm_R2-4-015_door | /light/middle| light         |
| luminary2      | lm_R2-4-015_door | /light/right | light         |
| Presence       | ps_R2-4-015_door | /ps          | p-sensor      |
| sensor         |                 |               |               |
+----------------+-----------------+---------------+---------------+
```

                  Table 4: Resource Directory identifiers

The CT inserts the end-points of the luminaries and the sensor in the
RD using the Context parameter (con) to specify the interface
address:

```
Req: POST coap://[FDFD::ABCD:0]/rd
  ?ep=lm_R2-4-015_wndw&con=coap://[FDFD::ABCD:1]
Payload:
</light/left>;rt="light";
  d="R2-4-015";ins="lamp4444";exp,
</light/middle>;rt="light";
  d="R2-4-015";ins="lamp5555";exp,
</light/right>;rt="light";
  d="R2-4-015";ins="lamp6666";exp

Res: 2.01 Created
Location: /rd/4521
```

```
Req: POST coap://[FDFD::ABCD:0]/rd
  ?ep=lm_R2-4-015_door&con=coap://[FDFD::ABCD:2]
Payload:
</light/left>;rt="light";
  d="R2-4-015";ins="lamp1111";exp,
</light/middle>;rt="light";
  d="R2-4-015";ins="lamp2222";exp,
</light/right>;rt="light";
  d="R2-4-015";ins="lamp3333";exp

Res: 2.01 Created
Location: /rd/4522

Req: POST coap://[FDFD::ABCD:0]/rd
  ?ep=ps_R2-4-015_door&con=coap://[FDFD::ABCD:3]
Payload:
</ps>;rt="p-sensor";
  d="R2-4-015";ins="pres1234";exp

Res: 2.01 Created
Location: /rd/4523
```

The domain name d="R2-4-015" has been added for an efficient lookup
because filtering on "ep" name is awkward.  The same domain name is
communicated to the two luminaries and the presence sensor by the CT.
The "exp" attribute is set for the later administration in DNS of the
instance name ins="lampxxxx".

Once the individual endpoints are registered, the group needs to be
registered.  Because the presence sensor sends one multicast message
to the luminaries, all lamps in the group need to have an identical
path.  This path is created on the two luminaries using the batch
command defined in [I-D.ietf-core-interfaces].  The path to a batch
of lamps is defined as: /light/grp1.  In the example below, two
endpoints are updated with an additional resource using the path
/light/grp1 on the two luminaries.

```
Req: POST
 coap://[FDFD::ABCD:1]/light/grp1
 (Content-Format:application/link-format)<light/middle>,<light/left>

Res: 2.04 Changed

Req: POST
 coap://[FDFD::ABCD:2]/light/grp1
(Content-Format:application/link-format)<light/right>

Res: 2.04 Changed
```

The group is specified in the RD.  The Context parameter is set to
the site-local multicast address allocated to the group.  In the POST
in the example below, these two end-points and the end-point of the
presence sensor are registered as members of the group.

It is expected that Standards Developing Organizations (SDOs) may
develop other special purpose protocols to specify additional group
links, group membership, group names and other parameters in the
individual nodes.

```
Req: POST coap://[FDFD::ABCD:0]/rd-group
?gp=grp_R2-4-015;con="coap//[FF05::1]";exp;ins="grp1234"
Payload:
<>ep=lm_R2-4-015_wndw,
<>ep=lm_R2-4-015_door,
<>ep=ps_R2-4-015_door


Res: 2.01 Created
Location: /rd-group/501
```

After the filling of the RD by the CT, the application in the
luminaries can learn to which groups they belong, and enable their
interface for the multicast address.

The luminary, knowing its domain, queries the RD for the end-point
with rt=light and d=R2-4-015.  The RD returns all end-points in the
domain.

```
Req: GET coap://[FDFD::ABCD:0]/rd-lookup/ep
  ?d=R2-4-015;rt=light

Res: 2.05 Content
<coap://[FDFD::ABCD:1]>;
  ep="lm_R2-4-015_wndw",
<coap://[FDFD::ABCD:2]>;
    ep="lm_R2-4-015_door"
```

Knowing its own IPv6 address, the luminary discovers its endpoint
name.  With the end-point name the luminary queries the RD for all
groups to which the end-point belongs.

```
Req: GET coap://[FDFD::ABCD:0]/rd-lookup/gp
  ?ep=lm_R2-4-015_wndw

Res: 2.05 Content
<coap://[FF05::1]>;gp="grp_R2-4-015"
```

From the context parameter value, the luminary learns the multicast address of the multicast group.

Alternatively, the CT can communicate the multicast address directly to the luminaries by using the "coap-group" resource specified in [RFC7390].

```
Req: POST //[FDFD::ABCD:1]/coap-group
        Content-Format: application/coap-group+json
     { "a": "[FF05::1]" }
     { "n": "grp_R2-4-015"}

Res: 2.01 Created
Location-Path: /coap-group/1
```

Dependent on the situation only the address ,"a", or the name, "n", is specified in the coap-group resource.  Instead of the RD group name also the DNS group name can be used.

12.1.3.  DNS entries

The network manager assigns the domain bc.example.com to the entries coming from the RD.  The agent that looks up the resource directory uses the domain name bc.example.com as prescribed, to enter the services and hosts into the DNS.

The agent does a lookup as specified in Section 9.6.  The RD returns all entries annotated with "exp".  The agent subsequently registers the following DNS-SD RRs:

```
   lm_R2-4-015_wndw.bc.example.com.        IN AAAA      FDFD::ABCD:1
   lm_R2-4-015_door.bc.example.com.        IN AAAA      FDFD::ABCD:2
   ps_R2-4-015_door.bc.example.com.        IN AAAA      FDFD::ABCD:3
   _light._udp.bc.example.com             IN PTR
                          lamp1111._light._udp.bc.example.com
   _light._udp.bc.example.com             IN PTR
                          lamp2222._light._udp.bc.example.com
   _light._udp.bc.example.com             IN PTR
                          lamp3333._light._udp.bc.example.com
   _light._udp.bc.example.com             IN PTR
                          lamp4444._light._udp.bc.example.com
   _light._udp.bc.example.com             IN PTR
                          lamp5555._light._udp.bc.example.com
   _light._udp.bc.example.com             IN PTR
                          lamp6666._light._udp.bc.example.com
   _p-sensor._udp.bc.example.com          IN PTR
                          pres12324._p-sensor._udp.bc.example.com
   lamp1111._light._udp.bc.example.com    IN SRV  0 0 5683
                          lm_R2-4-015_door.bc.example.com.
   lamp1111._light._udp.bc.example.com    IN TXT
                             txtver=1;path=/light/left
   lamp2222._light._udp.bc.example.com    IN SRV  0 0 5683
                          lm_R2-4-015_door.bc.example.com.
   lamp2222._light._udp.bc.example.com    IN TXT
                             txtver=1;path=/light/middle
   lamp3333._light._udp.bc.example.com    IN SRV  0 0 5683
                          lm_R2-4-015_door.bc.example.com.
   lamp3333._light._udp.bc.example.com    IN TXT
                             txtver=1;path=/light/right
   lamp4444._light._udp.bc.example.com    IN SRV  0 0 5683
                          lm_R2-4-015_wndw.bc.example.com.
   lamp4444._light._udp.bc.example.com    IN TXT
                             txtver=1;path=/light/left
   lamp5555._light._udp.bc.example.com    IN SRV  0 0 5683
                          lm_R2-4-015_wndw.bc.example.com.
   lamp5555._light._udp.bc.example.com    IN TXT
                             txtver=1;path=/light/middle
   lamp6666._light._udp.bc.example.com    IN SRV  0 0 5683
                          lm_R2-4-015_wndw.bc.example.com.
   lamp6666._light._udp.bc.example.com    IN TXT
                             txtver=1;path=/light/right
   pres1234._p-sensor._udp.bc.example.com IN SRV  0 0 5683
                          ps_R2-4-015_door.bc.example.com.
   pres1234._p-sensor._udp.bc.example.com IN TXT
                                 txtver=1;path=/ps
```

   To ask for all lamps is equivalent to returning all PTR RR with label
   _light.udp.bc.example.com.  from the DNS.  When it is required to

filter on the rd=R2-4-015 value in the DNS, additional PTR RRs have
to be entered into the DNS.

```
R2-4-015._light._udp.bc.example.com            IN PTR
                        lamp1111._light._udp.bc.example.com
R2-4-015._light._udp.bc.example.com            IN PTR
                        lamp2222._light._udp.bc.example.com
R2-4-015._light._udp.bc.example.com            IN PTR
                        lamp3333._light._udp.bc.example.com
R2-4-015._light._udp.bc.example.com            IN PTR
                        lamp4444._light._udp.bc.example.com
R2-4-015._light._udp.bc.example.com            IN PTR
                        lamp5555._light._udp.bc.example.com
R2-4-015._light._udp.bc.example.com            IN PTR
                        lamp6666._light._udp.bc.example.com
```

Returning all PTR RRs with label R2-4-015._light._udp.bc.example.com
provides all service instances within the domain R2-4-015.  This
filtering can be handy when there are many rooms.  In the example
there is only one room, making the filtering superfluous.

The agent can also discover groups that need to be discovered.  It
queries RD to return all groups which are exported.

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

   Res: 2.05 Content
   <coap://[FF05::1]/>;exp;gp="grp_R2-4-015;ins="grp1234";
ep="lm_R2-4-015_wndw";
ep="lm_R2-4-015_door
```

The group with FQDN grp_R2-4-015.bc.example.com can be entered into
the DNS by the agent.  The accompanying instance name is grp1234.
The <ServiceType> is chosen to be _group._udp.  The agent enters the
following RRs into the DNS.

```
grp_R2-4-015.bc.example.com.       IN AAAA           FF05::1
_group._udp.bc.example.com         IN PTR
                        grp1234._group._udp.bc.example.com
grp1234._group._udp.bc.example.com  IN SRV  0 0 5683
                        grp_R2-4-015_door.bc.example.com.
grp1234._group._udp.bc.example.com  IN TXT
                           txtver=1;path=/light/grp1
```

12.1.4.  RD Operation

   The specification of the group can be used by devices other than the
   luminaries and the sensor to learn the multicast address of the group
   in a given room.  For example a smart phone may be used to adjust the
   lamps in the room.

   After entry into the room, on request of the user, the smart phone
   queries the presence of RDs and may display all the domain names
   found on the RDs.  The user can, for example, scroll all domains
   (room names in this case) and select the room that he entered.  After
   selection the phone shows all groups in the selected room with their
   members.  Selecting a group, the user can dim, switch on/off the
   group of lights, or possibly even create temporary new groups.

   In all examples the SLAAC IPv6 address can be exchanged with the
   FQDN, when a connection to DNS exists.  Using the FQDN, a node learns
   the interface's IPv6 address, or the group's multicast address from
   DNS.  In the same way the presence sensor can learn the multicast
   address to which it should send its presence messages.

12.2.  OMA Lightweight M2M (LWM2M) Example

   This example shows how the OMA LWM2M specification makes use of
   Resource Directory (RD).

   OMA LWM2M is a profile for device services based on CoAP, CoRE RD,
   and other IETF RFCs and drafts.  LWM2M defines a simple object model
   and a number of abstract interfaces and operations for device
   management and device service enablement.

   An LWM2M server is an instance of an LWM2M middleware service layer,
   containing a Resource Directory along with other LWM2M interfaces
   defined by the LWM2M specification.

   CoRE Resource Directory (RD) is used to provide the LWM2M
   Registration interface.

   LWM2M does not provide for registration domains and does not
   currently use the rd-group or rd-lookup interfaces.

   The LWM2M specification describes a set of interfaces and a resource
   model used between a LWM2M device and an LWM2M server.  Other
   interfaces, proxies, applications, and function sets are currently
   out of scope for LWM2M.

   The location of the LWM2M Server and RD Function Set is provided by
   the LWM2M Bootstrap process, so no dynamic discovery of the RD

function set is used.  LWM2M Servers and endpoints are not required
to implement the ./well-known/core resource.

12.2.1.  The LWM2M Object Model

The OMA LWM2M object model is based on a simple 2 level class
hierarchy consisting of Objects and Resources.

An LWM2M Resource is a REST endpoint, allowed to be a single value or
an array of values of the same data type.

An LWM2M Object is a resource template and container type that
encapsulates a set of related resources.  An LWM2M Object represents
a specific type of information source; for example, there is a LWM2M
Device Management object that represents a network connection,
containing resources that represent individual properties like radio
signal strength.

Since there may potentially be more than one of a given type object,
for example more than one network connection, LWM2M defines instances
of objects that contain the resources that represent a specific
physical thing.

The URI template for LWM2M consists of a base URI followed by Object,
Instance, and Resource IDs:

{/base-uri}{/object-id}{/object-instance}{/resource-id}{/resource-
instance}

The five variables given here are strings.  base-uri can also have
the special value "undefined" (sometimes called "null" in RFC 6570).
Each of the variables object-instance, resource-id, and resource-
instance can be the special value "undefined" only if the values
behind it in this sequence also are "undefined".  As a special case,
object-instance can be "empty" (which is different from "undefined")
if resource-id is not "undefined".  [_TEMPLATE_TODO]

base-uri := Base URI for LWM2M resources or "undefined" for default
(empty) base URI

object-id := OMNA registered object ID (0-65535)

object-instance := Object instance identifier (0-65535) or
"undefined"/"empty" (see above)) to refer to all instances of an
object ID

resource-id := OMNA registered resource ID (0-65535) or "undefined"
to refer to all resources within an instance

resource-instance := Resource instance identifier or "undefined" to
refer to single instance of a resource

LWM2M IDs are 16 bit unsigned integers represented in decimal (no
leading zeroes except for the value 0) by URI format strings.  For
example, a LWM2M URI might be:

/1/0/1

The base uri is empty, the Object ID is 1, the instance ID is 0, the
resource ID is 1, and the resource instance is "undefined".  This
example URI points to internal resource 1, which represents the
registration lifetime configured, in instance 0 of a type 1 object
(LWM2M Server Object).

12.2.2.  LWM2M Register Endpoint

LWM2M defines a registration interface based on the Resource
Directory Function Set, described in Section 5.  The URI of the LWM2M
Resource Directory function set is specified to be "/rd" as
recommended in Section 5.2.

LWM2M endpoints register object IDs, for example </1>, to indicate
that a particular object type is supported, and register object
instances, for example </1/0>, to indicate that a particular instance
of that object type exists.

Resources within the LWM2M object instance are not registered with
the RD, but may be discovered by reading the resource links from the
object instance using GET with a CoAP Content-Format of application/
link-format.  Resources may also be read as a structured object by
performing a GET to the object instance with a Content-Format of
senml+json.

When an LWM2M object or instance is registered, this indicates to the
LWM2M server that the object and it's resources are available for
management and service enablement (REST API) operations.

LWM2M endpoints may use the following RD registration parameters as
defined in Table 1 :

ep - Endpoint Name
lt - registration lifetime

Endpoint Name is mandatory, all other registration parameters are
optional.

Additional optional LWM2M registration parameters are defined:

```
+-----------+-------+----------------------------+-------------+
| Name      | Query | Validity                   | Description |
+-----------+-------+----------------------------+-------------+
| Protocol  | b     | {"U","UQ","S","SQ","US","UQS"} | Available   |
| Binding   |       |                            | Protocols   |
|           |       |                            |             |
| LWM2M     | ver   | 1.0                        | Spec Version |
| Version   |       |                            |             |
|           |       |                            |             |
| SMS Number | sms  |                            | MSISDN      |
+-----------+-------+----------------------------+-------------+
```

              Table 5: LWM2M Additional Registration Parameters

   The following RD registration parameters are not currently specified
   for use in LWM2M:

   et - Endpoint Type
   con - Context

   The endpoint registration must include a payload containing links to
   all supported objects and existing object instances, optionally
   including the appropriate link-format relations.

   Here is an example LWM2M registration payload:

   </1>,</1/0>,</3/0>,</5>

   This link format payload indicates that object ID 1 (LWM2M Server
   Object) is supported, with a single instance 0 existing, object ID 3
   (LWM2M Device object) is supported, with a single instance 0
   existing, and object 5 (LWM2M Firmware Object) is supported, with no
   existing instances.

12.2.3.  Alternate Base URI

   If the LWM2M endpoint exposes objects at a base URI other than the
   default empty base path, the endpoint must register the base URI
   using rt="oma.lwm2m".  An example link payload using alternate base
   URI would be:

   </my_lwm2m>;rt="oma.lwm2m",</my_lwm2m/1>,<my_lwm2m/1/0>,<my_lwm2m/5>

   This link payload indicates that the lwm2m objects will be placed
   under the base URI "/my_lwm2m" and that object ID 1 (server) is
   supported, with a single instance 0 existing, and object 5 (firmware
   update) is supported.

12.2.4.  LWM2M Update Endpoint Registration

   An LWM2M Registration update proceeds as described in Section 5.3,
   and adds some optional parameter updates:

   lt - Registration Lifetime
   b - Protocol Binding
   sms - MSISDN
   link payload - new or modified links

   A Registration update is also specified to be used to update the
   LWM2M server whenever the endpoint's UDP port or IP address are
   changed.

12.2.5.  LWM2M De-Register Endpoint

   LWM2M allows for de-registration using the delete method on the
   returned location from the initial registration operation.  LWM2M de-
   registration proceeds as described in Section 5.4.

13.  Acknowledgments

   Srdjan Krco, Szymon Sasin, Kerry Lynn, Esko Dijk, Anders Brandt,
   Matthieu Vial, Mohit Sethi, Sampo Ukkola and Linyi Tian have provided
   helpful comments, discussions and ideas to improve and shape this
   document.  Section 9 is based on an earlier draft by Kerry Lynn.
   Zach would also like to thank his colleagues from the EU FP7 SENSEI
   project, where many of the resource directory concepts were
   originally developed.

14.  Changelog

   changes from -06 to -07

   o  added text in the discovery section to allow content format hints
      to be exposed in the discovery link attributes

   o  editorial updates to section 9

   o  update author information

   o  minor text corrections

   Changes from -05 to -06

   o  added note that the PATCH section is contingent on the progress of
      the PATCH method

changes from -04 to -05

o  added Update Endpoint Links using PATCH

o  http access made explicit in interface specification

o  Added http examples

Changes from -03 to -04:

o  Added http response codes

o  Clarified endpoint name usage

o  Add application/link-format+cbor content-format

Changes from -02 to -03:

o  Added an example for lighting and DNS integration

o  Added an example for RD use in OMA LWM2M

o  Added Read Links operation for link inspection by endpoints

o  Expanded DNS-SD section

o  Added draft authors Peter van der Stok and Michael Koster

Changes from -01 to -02:

o  Added a catalogue use case.

o  Changed the registration update to a POST with optional link
   format payload.  Removed the endpoint type update from the update.

o  Additional examples section added for more complex use cases.

o  New DNS-SD mapping section.

o  Added text on endpoint identification and authentication.

o  Error code 4.04 added to Registration Update and Delete requests.

o  Made 63 bytes a SHOULD rather than a MUST for endpoint name and
   resource type parameters.

Changes from -00 to -01:

o  Removed the ETag validation feature.

o  Place holder for the DNS-SD mapping section.

o  Explicitly disabled GET or POST on returned Location.

o  New registry for RD parameters.

o  Added support for the JSON Link Format.

o  Added reference to the Groupcomm WG draft.

Changes from -05 to WG Document -00:

o  Updated the version and date.

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 and 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.

o  Fixed tickets 383 and 372

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= parameter to be present.

Changes from -01 to -02:

o  Added a terminology section.

o  Changed the inclusion 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.

15.  References

15.1.  Normative References

[I-D.ietf-core-links-json]
          Li, K., Rahman, A., and C. Bormann, "Representing CoRE
          Formats in JSON and CBOR", draft-ietf-core-links-json-04
          (work in progress), November 2015.

[RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986]  Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
          Resource Identifier (URI): Generic Syntax", STD 66,
          RFC 3986, DOI 10.17487/RFC3986, January 2005,
          <http://www.rfc-editor.org/info/rfc3986>.

[RFC5226]  Narten, T. and H. Alvestrand, "Guidelines for Writing an
          IANA Considerations Section in RFCs", BCP 26, RFC 5226,
          DOI 10.17487/RFC5226, May 2008,
          <http://www.rfc-editor.org/info/rfc5226>.

   [RFC5988]  Nottingham, M., "Web Linking", RFC 5988,
              DOI 10.17487/RFC5988, October 2010,
              <http://www.rfc-editor.org/info/rfc5988>.

   [RFC6335]  Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S.
              Cheshire, "Internet Assigned Numbers Authority (IANA)
              Procedures for the Management of the Service Name and
              Transport Protocol Port Number Registry", BCP 165,
              RFC 6335, DOI 10.17487/RFC6335, August 2011,
              <http://www.rfc-editor.org/info/rfc6335>.

   [RFC6570]  Gregorio, J., Fielding, R., Hadley, M., Nottingham, M.,
              and D. Orchard, "URI Template", RFC 6570,
              DOI 10.17487/RFC6570, March 2012,
              <http://www.rfc-editor.org/info/rfc6570>.

   [RFC6690]  Shelby, Z., "Constrained RESTful Environments (CoRE) Link
              Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
              <http://www.rfc-editor.org/info/rfc6690>.

   [RFC6763]  Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <http://www.rfc-editor.org/info/rfc6763>.

   [RFC7396]  Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396,
              DOI 10.17487/RFC7396, October 2014,
              <http://www.rfc-editor.org/info/rfc7396>.

15.2.  Informative References

   [I-D.ietf-core-interfaces]
              Shelby, Z., Vial, M., and M. Koster, "Reusable Interface
              Definitions for Constrained RESTful Environments", draft-
              ietf-core-interfaces-04 (work in progress), October 2015.

   [RFC1034]  Mockapetris, P., "Domain names - concepts and facilities",
              STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987,
              <http://www.rfc-editor.org/info/rfc1034>.

   [RFC1123]  Braden, R., Ed., "Requirements for Internet Hosts -
              Application and Support", STD 3, RFC 1123,
              DOI 10.17487/RFC1123, October 1989,
              <http://www.rfc-editor.org/info/rfc1123>.

   [RFC3629]  Yergeau, F., "UTF-8, a transformation format of ISO
              10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November
              2003, <http://www.rfc-editor.org/info/rfc3629>.

   [RFC5198]  Klensin, J. and M. Padlipsky, "Unicode Format for Network
              Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008,
              <http://www.rfc-editor.org/info/rfc5198>.

   [RFC6775]  Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C.
              Bormann, "Neighbor Discovery Optimization for IPv6 over
              Low-Power Wireless Personal Area Networks (6LoWPANs)",
              RFC 6775, DOI 10.17487/RFC6775, November 2012,
              <http://www.rfc-editor.org/info/rfc6775>.

   [RFC7230]  Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Message Syntax and Routing",
              RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <http://www.rfc-editor.org/info/rfc7230>.

   [RFC7252]  Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
              Application Protocol (CoAP)", RFC 7252,
              DOI 10.17487/RFC7252, June 2014,
              <http://www.rfc-editor.org/info/rfc7252>.

   [RFC7390]  Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for
              the Constrained Application Protocol (CoAP)", RFC 7390,
              DOI 10.17487/RFC7390, October 2014,
              <http://www.rfc-editor.org/info/rfc7390>.

Editorial Comments

[_TEMPLATE_TODO] This text needs some help from an RFC 6570 expert.

Authors' Addresses

   Zach Shelby
   ARM
   150 Rose Orchard
   San Jose  95134
   USA

   Phone: +1-408-203-9434
   Email: zach.shelby@arm.com

Michael Koster
SmartThings
665 Clyde Avenue
Mountain View  94043
USA

Phone: +1-707-502-5136
Email: Michael.Koster@smartthings.com


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

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


Peter van der Stok
consultant

Phone: +31-492474673 (Netherlands), +33-966015248 (France)
Email: consultancy@vanderstok.org
URI:   www.vanderstok.org