CoRE   Z. Shelby
Internet-Draft   ARM
Intended status: Standards Track   M. Koster
Expires: April 6, 2019   SmartThings
                                                           C. Bormann
                                              Universitaet Bremen TZI
                                                     P. van der Stok
                                                          consultant
                                                     C. Amsuess, Ed.
                                                    October 03, 2018

CoRE Resource Directory
draft-ietf-core-resource-directory-15

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
   registrations 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 for web servers to
   discover the RD and to register, maintain, lookup and remove resource
   descriptions. Furthermore, new link attributes useful in conjunction
   with an RD are defined.

Status of This Memo

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 querying "/.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 registrations 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 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 [RFC3986], [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:

resolve against
   The expression "a URI-reference is _resolved against_ a base URI"
   is used to describe the process of [RFC3986] Section 5.2.
   Noteworthy corner cases are that if the URI-reference is a (full)

URI and resolved against any base URI, that gives the original
full URI, and that resolving an empty URI reference gives the base
URI without any fragment identifier.

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.

Sector
    In the context of a Resource Directory, a sector is a logical
    grouping of endpoints.

    The abbreviation "d=" is used for the sector in query parameters
    for compatibility with deployed implementations.

Group
    A group in the Resource Directory specifies a set of endpoints
    that are enabled with the same multicast address for the purpose
    of efficient group communications. All groups within a sector
    have unique names.

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 has a unique name
    within the associated sector of the registration.

Registration Base URI
    The Base URI of a Registration is a URI that typically gives
    scheme and authority information about an Endpoint. The
    Registration Base URI is provided at registration time, and is
    used by the Resource Directory to resolve relative references of
    the registration into URIs.

Target
    The target of a link is the destination address (URI) of the link.
    It is sometimes identified with "href=", or displayed as
    "<target>". Relative targets need resolving with respect to the
    Base URI (section 5.2 of [RFC3986]).

    This use of the term Target is consistent with [RFC8288]'s use of
    the term.

Context

   The context of a link is the source address (URI) of the link, and
   describes which resource is linked to the target. A link's
   context is made explicit in serialized links as the "anchor="
   attribute.

   This use of the term Context is consistent with [RFC8288]'s use of
   the term.

Directory Resource
   A resource in the Resource Directory (RD) containing registration
   resources.

Group Resource
   A resource in the RD containing registration resources of the
   Endpoints that form a group.

Registration Resource
   A resource in the RD that contains information about an Endpoint
   and its links.

Commissioning Tool
   Commissioning Tool (CT) is a device that assists during the
   installation of the network by assigning values to parameters,
   naming endpoints and groups, or adapting the installation to the
   needs of the applications.

Registrant-ep
   Registrant-ep is the endpoint that is registered into the RD. The
   registrant-ep can register itself, or a CT registers the
   registrant-ep.

RDAO
   Resource Directory Address Option.

For several operations, interface descriptions are given in list
form; those describe the operation participants, request codes, URIs,
content formats and outcomes. Those templates contain normative
content in their Interaction, Method, URI Template and URI Template
Variables sections as well as the details of the Success condition.
The additional sections on options like Content-Format and on Failure
codes give typical cases that an implementation of the RD should deal
with. Those serve to illustrate the typical responses to readers who
are not yet familiar with all the details of CoAP based interfaces;
they do not limit what a server may respond under atypical
circumstances.

3. Architecture and Use Cases

3.1. Principles

The Resource Directory is primarily a tool to make discovery
operations more efficient than querying /.well-known/core on all
connected devices, or across boundaries that would be limiting those
operations.

It provides a cache (in the high-level sense, not as defined in
[RFC7252]/[RFC2616]) of data that could otherwise only be obtained by
directly querying the /.well-known/core resource on the target
device, or by accessing those resources with a multicast request.

Only information SHOULD be stored in the resource directory that is
discoverable from querying the described device's /.well-known/core
resource directly.

Data in the resource directory can only be provided by the device
which hosts those data or a dedicated Commissioning Tool (CT). These
CTs are thought to act on behalf of endpoints too constrained, or
generally unable, to present that information themselves. No other
client can modify data in the resource directory. Changes in the
Resource Directory do not propagate automatically back to the web
server from where the links originated.

3.2. Architecture

The resource directory architecture is illustrated in Figure 1. A
Resource Directory (RD) is used as a repository for Web Links
[RFC5988] describing resources hosted on other web servers, also
called endpoints (EP). An endpoint is a web server associated with a
scheme, IP address and port. 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
registration entries), and for endpoints to lookup resources from the
RD or maintain groups. An RD can be logically segmented by the use
of Sectors. The set of endpoints grouped for group communication can
be defined by the RD or configured by a Commissioning Tool. This
information hierarchy is shown in Figure 2.

A mechanism to discover an RD using CoRE Link Format [RFC6690] is
defined.

Registration entries in the RD are soft state and need to be
periodically refreshed.

An endpoint uses specific interfaces to register, update and remove a
resource directory registration entry. It is also possible for an RD
to fetch Web Links from endpoints and add them as resource directory
registration entries.

At the first registration of a set of entries, a "registration
resource" is created, the location of which is returned to the
registering endpoint. The registering endpoint uses this
registration resource to manage the contents of registration 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 |----      |                 |
  +----+
```
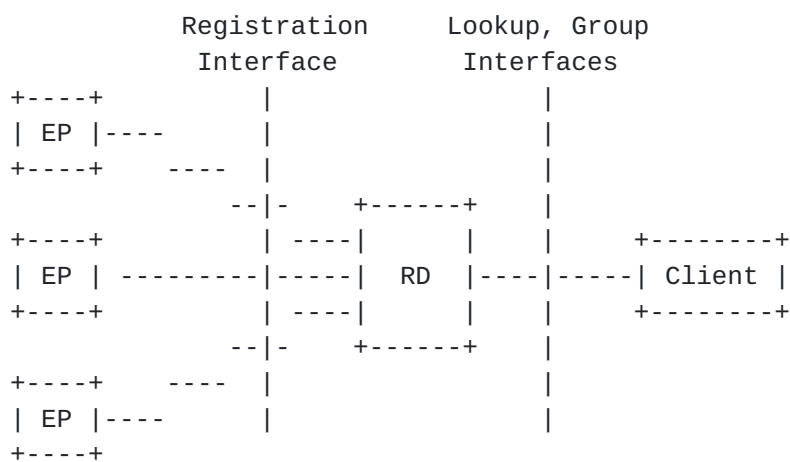
Figure 1: The resource directory architecture.

```
        +------------+
        |   Group    | <-- Name, Scheme, IP, Port
        +------------+
             |
             |
        +------------+
        |  Endpoint  | <-- Name, Scheme, IP, Port
        +------------+
             |
             |
        +------------+
        |  Resource  | <-- Target, Parameters
        +------------+
```
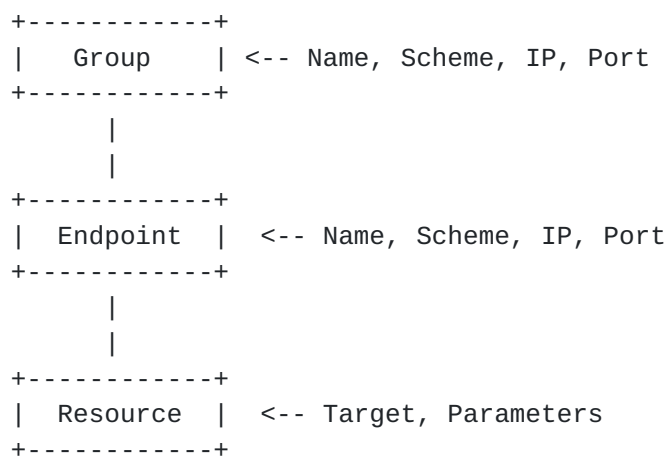
Figure 2: The resource directory information hierarchy.

3.3. RD Content Model

   The Entity-Relationship (ER) models shown in Figure 3 and Figure 4
   model the contents of /.well-known/core and the resource directory
   respectively, with entity-relationship diagrams [ER]. Entities
   (rectangles) are used for concepts that exist independently.
   Attributes (ovals) are used for concepts that exist only in
   connection with a related entity. Relations (diamonds) give a
   semantic meaning to the relation between entities. Numbers specify
   the cardinality of the relations.

   Some of the attribute values are URIs. Those values are always full
   URIs and never relative references in the information model. They
   can, however, be expressed as relative references in serializations,
   and often are.

   These models provide an abstract view of the information expressed in
   link-format documents and a Resource Directory. They cover the
   concepts, but not necessarily all details of an RD's operation; they
   are meant to give an overview, and not be a template for
   implementations.

```
                    +----------------------+
                    |   /.well-known/core  |
                    +----------------------+
                               |
                               | 1
                        ////////\\\\\\\
                        <     contains    >
                         \\\\\\\\///////
                               |
                               | 0+
                    +-------------------+
                    |       link        |
                    +-------------------+
                               |
                               |  1   oooooooo
                               +-----o target o
                               |        oooooooo
         oooooooooooo    0+    |
         o     target  o--------+
         o  attribute o        | 0+   oooooo
          oooooooooooo         +-----o rel  o
                               |        oooooo
                               |
                               | 1    oooooooooo
                               +-----o context o
                                        oooooooooo
```
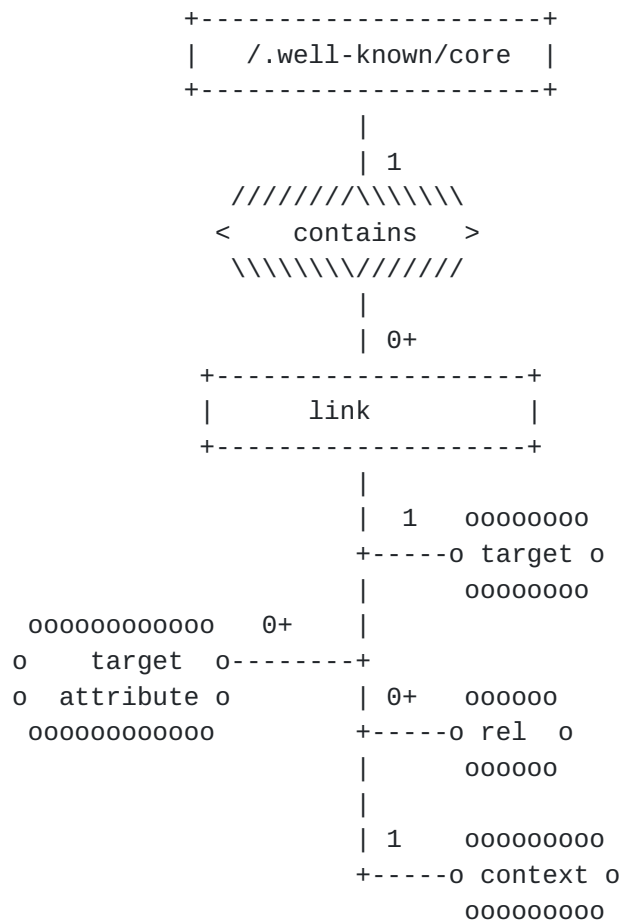
Figure 3: E-R Model of the content of /.well-known/core

The model shown in Figure 3 models the contents of /.well-known/core
which contains:

o a set of links belonging to the hosting web server

The web server is free to choose links it deems appropriate to be
exposed in its ".well-known/core". Typically, the links describe
resources that are served by the host, but the set can also contain
links to resources on other servers (see examples in [RFC6690] page
14). The set does not necessarily contain links to all resources
served by the host.

A link has the following attributes (see [RFC5988]):

o Zero or more link relations: They describe relations between the
   link context and the link target.

In link-format serialization, they are expressed as space-
separated values in the "rel" attribute, and default to "hosts".

o A link context URI: It defines the source of the relation, e.g.
   who "hosts" something.

   In link-format serialization, it is expressed in the "anchor"
   attribute. It defaults to that document's URI.

o A link target URI: It defines the destination of the relation
   (e.g. _what_ is hosted), and is the topic of all target
   attributes.

   In link-format serialization, it is expressed between angular
   brackets, and sometimes called the "href".

o Other target attributes (e.g. resource type (rt), interface (if),
   or content-type (ct)). These provide additional information about
   the target URI.

```
                 +----------------------+           1  ooooooo
                 |  resource-directory  |          +--o  href o
                 +----------------------+          |   ooooooo
                        | 1                        |
                        |          oooooooooo  0-1 |       1  oooooo
                        |            o    base  o---+ | +------o  gp  o
                        |          oooooooooo     | | |        oooooo
                        |                          | | |
                  //////\\\\              0+ +--------+  0-1  ooooo
                 < contains >---------------| group  |------o  d  o
                  \\\\\/////               +--------+        ooooo
                        |                        | 0+
                  0+ |                           |
    ooooooo     1  +---------------+  1+      //////\\\\\\
   o  base o-------|  registration |---------< composed of >
    ooooooo        +---------------+          \\\\\\/////
                      |        | 1
                      |        +--------------+
      oooooooo    1 |                         |
       o  href  o----+              //////\\\\
      oooooooo       |             < contains >
                     |              \\\\\/////
      oooooooo    1 |                    |
       o   ep   o----+                   | 0+
      oooooooo       |         +-----------------+
                     |         |      link       |
      oooooooo 0-1 |         +-----------------+
       o    d   o----+                   |
      oooooooo       |                   | 1   oooooooo
                     |                   +-----o target o
      oooooooo    1 |                   |       oooooooo
       o   lt   o----+    oooooooooo    0+ |
      oooooooo       |    o  target   o-----+
                     |    o attribute o    | 0+  oooooo
     oooooooooo 0+ |    oooooooooo      +-----o rel  o
      o  endpoint o----+                   |       oooooo
      o attribute o                        |
      oooooooooo                           | 1   ooooooooo
                                           +----o context o
                                                  ooooooooo
```
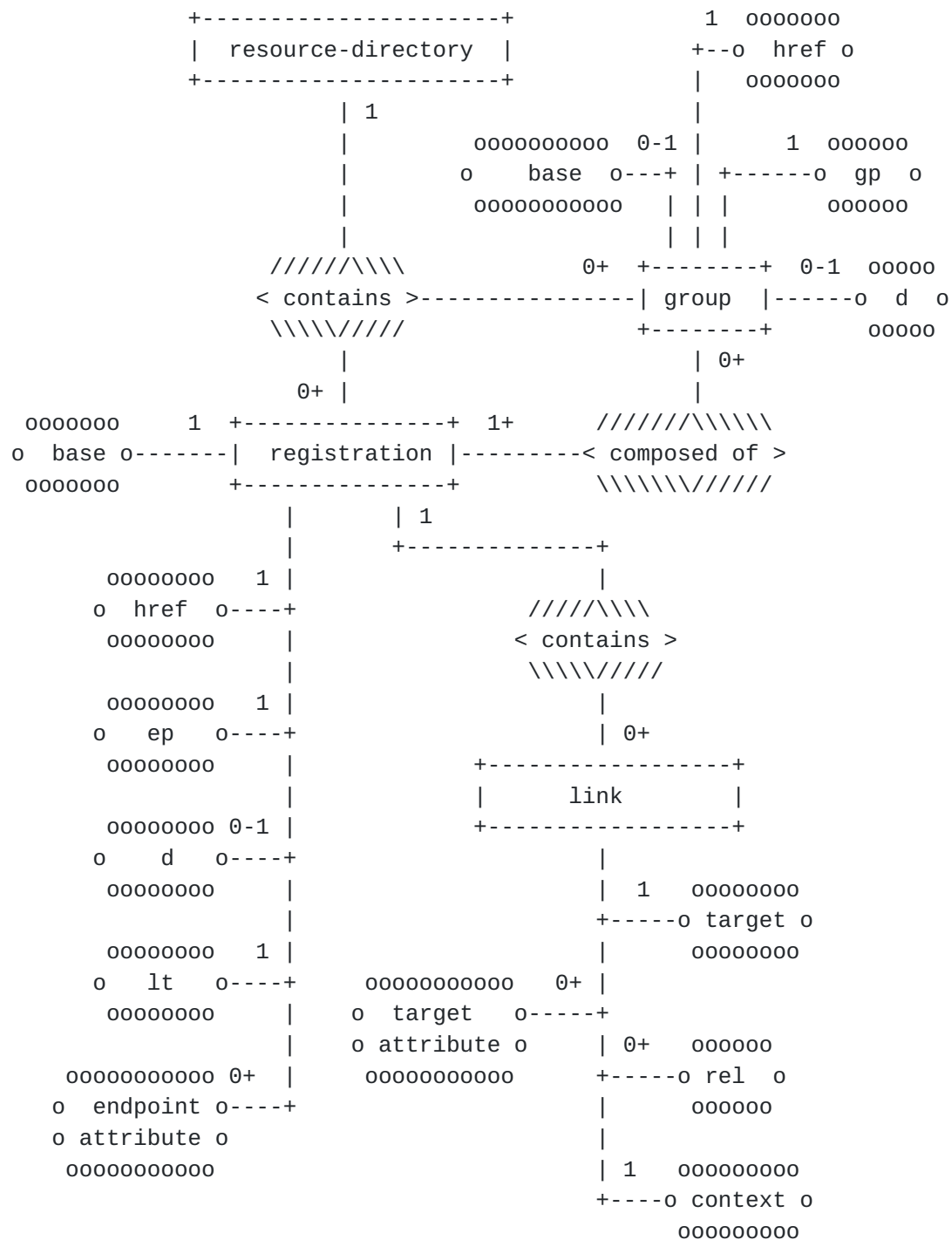
          Figure 4: E-R Model of the content of the Resource Directory

   The model shown in Figure 4 models the contents of the resource
   directory which contains in addition to /.well-known/core:

   o 0 to n Registration (entries) of endpoints,

o 0 or more Groups

A Group has:

o a group name ("gp"),

o optionally a sector (abbreviated "d" for historical reasons),

o a group resource location inside the RD ("href"),

o zero or one multicast addresses expressed as a base URI ("base"),

o and is composed of zero or more registrations (endpoints).

A registration is associated with one endpoint. A registration can
be part of 0 or more Groups . A registration defines a set of links
as defined for /.well-known/core. A Registration has six types of
attributes:

o a unique endpoint name ("ep") within a sector

o a Registration Base URI ("base", a URI typically describing the
   scheme://authority part)

o a lifetime ("lt"),

o a registration resource location inside the RD ("href"),

o optionally a sector ("d")

o optional additional endpoint attributes (from Section 10.3)

The cardinality of "base" is currently 1; future documents are
invited to extend the RD specification to support multiple values
(e.g. [I-D.silverajan-core-coap-protocol-negotiation]). Its value
is used as a Base URI when resolving URIs in the links contained in
the endpoint.

Links are modelled as they are in Figure 3.

3.4. 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
wireless 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.

Imagine a scenario where endpoints installed on vehicles enable
tracking of the position of these vehicles for fleet management
purposes and allow monitoring of environment parameters. During the
boot-up process endpoints register with a Resource Directory, which
is hosted by the mobile operator or somewhere in the cloud.
Periodically, these endpoints update their registration and may
modify resources they offer.

When endpoints are not always connected, for example because they
enter a sleep mode, a remote server is usually used to provide proxy
access to the endpoints. Mobile apps or web applications for
environment monitoring contact the RD, look up the endpoints capable
of providing information about the environment using an 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.

[3.5](). 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.

[3.6](). 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 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 RD Lookup 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 formats like [RFC6690] which may be internally
stored as triples, or relation/attribute pairs providing metadata
about resource links, need to be supported by Resource Directories .
External catalogues 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 sectors 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. Groups may be defined to support
efficient data transport.

4. Finding a Resource Directory

A (re-)starting device may want to find one or more resource
directories for discovery purposes.

The device may be pre-configured to exercise specific mechanisms for
finding the resource directory:

1. It may be configured with a specific IP address for the RD. That
    IP address may also be an anycast address, allowing the network
    to forward RD requests to an RD that is topologically close; each
    target network environment in which some of these preconfigured
    nodes are to be brought up is then configured with a route for
    this anycast address that leads to an appropriate RD. (Instead
    of using an anycast address, a multicast address can also be
    preconfigured. The RD servers then need to configure one of
    their interfaces with this multicast address.)

2. It may be configured with a DNS name for the RD and use DNS to
    return the IP address of the RD; it can find a DNS server to

perform the lookup using the usual mechanisms for finding DNS
servers.

3. It may be configured to use a service discovery mechanism such as
    DNS-SD [RFC6763]. The present specification suggests configuring
    the service with name rd._sub._coap._udp, preferably within the
    domain of the querying nodes.

For cases where the device is not specifically configured with a way
to find a resource directory, the network may want to provide a
suitable default.

1. If the address configuration of the network is performed via
    SLAAC, this is provided by the RDAO option Section 4.1.

2. If the address configuration of the network is performed via
    DHCP, this could be provided via a DHCP option (no such option is
    defined at the time of writing).

Finally, if neither the device nor the network offers any specific
configuration, the device may want to employ heuristics to find a
suitable resource directory.

The present specification does not fully define these heuristics, but
suggests a number of candidates:

1. In a 6LoWPAN, just assume the Border Router (6LBR) can act as a
    resource directory (using the ABRO option to find that
    [RFC6775]). Confirmation can be obtained by sending a Unicast to
    "coap://[6LBR]/.well-known/core?rt=core.rd*".

2. In a network that supports multicast well, discovering the RD
    using a multicast query for /.well-known/core as specified in
    CoRE Link Format [RFC6690]: Sending a Multicast GET to
    "coap://[MCD1]/.well-known/core?rt=core.rd*". RDs within the
    multicast scope will answer the query.

As some of the RD addresses obtained by the methods listed here 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.

If multiple candidate addresses are discovered, the device may pick
any of them initially, unless the discovery method indicates a more
precise selection scheme.

4.1. Resource Directory Address Option (RDAO)

The Resource Directory Address Option (RDAO) using IPv6 Neighbor
Discovery (ND) carries information about the address of the Resource
Directory (RD). This information is needed when endpoints cannot
discover the Resource Directory with a link-local or realm-local
scope multicast address because the endpoint and the RD are separated
by a Border Router (6LBR). In many circumstances the availability of
DHCP cannot be guaranteed either during commissioning of the network.
The presence and the use of the RD is essential during commissioning.

It is possible to send multiple RDAO options in one message,
indicating as many resource directory addresses.

The RDAO format is:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |     Type      |  Length = 3   |        Valid Lifetime         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                           Reserved                            |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  +                                                               +
  |                                                               |
  +                         RD Address                            +
  |                                                               |
  +                                                               +
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Fields:

   Type: 38

   Length: 8-bit unsigned integer. The length of
                          the option in units of 8 bytes.
                          Always 3.

   Valid Lifetime: 16-bit unsigned integer. The length of
                          time in units of 60 seconds (relative to
                          the time the packet is received) that
                          this Resource Directory address is valid.
                          A value of all zero bits (0x0) indicates
                          that this Resource Directory address
                          is not valid anymore.

   Reserved: This field is unused. It MUST be
                          initialized to zero by the sender and
                          MUST be ignored by the receiver.

   RD Address: IPv6 address of the RD.

                Figure 5: Resource Directory Address Option

[5](#). Resource Directory

   This section defines the required set of REST interfaces between a
   Resource Directory (RD) and endpoints. 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.

All operations on the contents of the Resource Directory MUST be atomic and idempotent.

A resource directory MAY make the information submitted to it 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.

5.1. Payload Content Formats

Resource Directory implementations using this specification MUST support the application/link-format content format (ct=40).

Resource Directories implementing this specification MAY support additional content formats.

Any additional content format supported by a Resource Directory implementing this specification MUST have an equivalent serialization in the application/link-format content format.

5.2. URI Discovery

Before an endpoint can make use of an RD, it must first know the RD's address and port, and the URI path information for its REST APIs. This section defines discovery of the RD and its URIs using the well-known interface of the CoRE Link Format [RFC6690]. A complete set of RD discovery methods is described in Section 4.

Discovery of the RD registration URI path 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 URIs for RD Lookup operations, core.rd* is used to discover all URI paths for RD operations, and "core.rd-group" is used to discover the URI path for RD Group operations. Upon success, the response will contain a payload with a link format entry for each RD function discovered, indicating the URI of the RD function returned and the corresponding Resource Type. When performing multicast discovery, the multicast IP address used will depend on the scope required and the multicast capabilities of the network (see Section 10.5.

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. The well-known entry points
SHOULD be provided to enable unicast discovery.

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

While the link targets in this discovery step are often expressed in
path-absolute form, this is not a requirement. Clients of the RD
SHOULD therefore accept URIs of all schemes they support, both as
URIs and relative references, and not limit the set of discovered
URIs to those hosted at the address used for URI discovery.

The URI Discovery operation can yield multiple URIs of a given
resource type. The client of the RD can use any of the discovered
addresses initially.

The discovery request interface is specified as follows (this is
exactly the Well-Known Interface of [RFC6690] Section 4, with the
additional requirement that the server MUST support query filtering):

Interaction: EP and Client -> RD

Method: GET

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

URI Template Variables:

   rt := Resource Type. SHOULD contain one of the values "core.rd",
      "core.rd-lookup*", "core.rd-lookup-res", "core.rd-lookup-ep",
      "core.rd-lookup-gp", "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" or 200 "OK" 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: 4.00 "Bad Request" or 400 "Bad Request" is returned in case
of a malformed request for a unicast request.

Failure: No error response to a multicast request.

HTTP support : YES (Unicast only)

The following example shows an endpoint discovering an RD using this
interface, thus learning that the directory resource location, in
this example, is /rd, and that the content-format delivered by the
server hosting the resource is application/link-format (ct=40). Note
that it is up to the RD to choose its RD locations.

Req: GET coap://[MCD1]/.well-known/core?rt=core.rd*

Res: 2.05 Content
</rd>;rt="core.rd";ct=40,
</rd-lookup/ep>;rt="core.rd-lookup-ep";ct=40,
</rd-lookup/res>;rt="core.rd-lookup-res";ct=40,
</rd-lookup/gp>;rt="core.rd-lookup-gp";ct=40,
</rd-group>;rt="core.rd-group";ct=40

Figure 6: Example discovery exchange

The following example shows the way of indicating that a client may
request alternate content-formats. The Content-Format code attribute
"ct" MAY include a space-separated sequence of Content-Format codes
as specified in Section 7.2.1 of [RFC7252], indicating that multiple
content-formats are available. The example below shows the required
Content-Format 40 (application/link-format) indicated as well as the
CBOR and JSON representation of link format. The RD resource
locations /rd, /rd-lookup, and /rd-group are example values. The
server in this example also indicates that it is capable of providing
observation on resource lookups.

[The RFC editor is asked to replace these and later occurrences of
MCD1, TBD64 and TBD504 with the numeric ID values assigned by IANA to
application/link-format+cbor and application/link-format+json,
respectively, as they are defined in I-D.ietf-core-links-json.]

```
Req: GET coap://[MCD1]/.well-known/core?rt=core.rd*

Res: 2.05 Content
</rd>;rt="core.rd";ct="40 65225",
</rd-lookup/res>;rt="core.rd-lookup-res";ct="40 TBD64 TBD504";obs,
</rd-lookup/ep>;rt="core.rd-lookup-ep";ct="40 TBD64 TBD504",
</rd-lookup/gp>;rt="core.rd-lookup-gp";ct=40 TBD64 TBD504",
</rd-group>;rt="core.rd-group";ct="40 TBD64 TBD504"
```

From a management and maintenance perspective, it is necessary to identify the components that constitute the RD server. The identification refers to information about for example client-server incompatibilities, supported features, required updates and other aspects. The URI discovery address, a described in section 4 of [RFC6690] can be used to find the identification.

It would typically be stored in an implementation information link (as described in [I-D.bormann-t2trg-rel-impl]):

```
Req: GET /.well-known/core?rel=impl-info

Res: 2.05 Content
<http://software.example.com/shiny-resource-directory/1.0beta1>;
    rel="impl-info"
```

Note that depending on the particular server's architecture, such a link could be anchored at the RD server's root, at the discovery site (as in this example) or at individual RD components. The latter is to be expected when different applications are run on the same server.

5.3. Registration

After discovering the location of an RD, a registrant-ep or CT MAY register the resources of the registrant-ep 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 parameters indicating the name of the endpoint, and optionally the sector, lifetime and base URI of the registration. It is expected that other specifications will define further parameters (see Section 10.3). The RD then creates a new registration resource in the RD and returns its location. The receiving endpoint MUST use that location when refreshing registrations using this interface. Registration resources in the RD are kept active for the period indicated by the lifetime parameter. The creating endpoint is

responsible for refreshing the registration resource 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 parameters ep and d (sector)
does not create multiple registration resources.

The following rules apply for an update identified by a given (ep, d)
value pair:

o when the parameter values of the Update generate the same
   attribute values as already present, the location of the already
   existing registration is returned.

o when for a given (ep, d) value pair the update generates attribute
   values which are different from the existing one, the existing
   registration is removed and a new registration with a new location
   is created.

o when the (ep, d) value pair of the update is different from any
   existing registration, a new registration is generated.

The posted link-format document can (and typically does) contain
relative references both in its link targets and in its anchors, or
contain empty anchors. The RD server needs to resolve these
references in order to faithfully represent them in lookups. They
are resolved against the base URI of the registration, which is
provided either explicitly in the "base" parameter or constructed
implicitly from the requester's URI as constructed from its network
address and scheme.

Link format documents submitted to the resource directory are
interpreted as Modernized Link Format (see Appendix D) by the RD. A
registrant-ep SHOULD NOT submit documents whose interpretations
according to [RFC6690] and Appendix D differ to avoid the ambiguities
described in Appendix B.4.

In practice, most links (precisely listed in Appendix D.1) can be
submitted without consideration for those details.

The registration request interface is specified as follows:

Interaction: EP -> RD

Method: POST

URI Template: {+rd}{?ep,d,lt,base,extra-attrs*}

URI Template Variables:

rd := RD registration URI (mandatory). This is the location of
   the RD, as obtained from discovery.

ep := Endpoint name (mostly mandatory). The endpoint name is an
   identifier that MUST be unique within a sector. The maximum
   length of this parameter is 63 bytes. If the RD is configured
   to recognize the endpoint (e.g. based on its security context),
   the endpoint sets no endpoint name, and the RD assigns one
   based on a set of configuration parameter values.

d := Sector (optional). The sector to which this endpoint
   belongs. The maximum length of this parameter is 63 bytes.
   When this parameter is not present, the RD MAY associate the
   endpoint with a configured default sector or leave it empty.
   The endpoint name and sector name are not set when one or both
   are set in an accompanying authorization token.

lt := Lifetime (optional). Lifetime of the registration in
   seconds. Range of 60-4294967295. If no lifetime is included
   in the initial registration, a default value of 90000 (25
   hours) SHOULD be assumed.

base := Base URI (optional). This parameter sets the base URI of
   the registration, under which the relative links in the payload
   are to be interpreted. The specified URI typically does not
   have a path component of its own, and MUST be suitable as a
   base URI to resolve any relative references given in the
   registration. The parameter is therefore usually of the shape
   "scheme://authority" for HTTP and CoAP URIs. The URI SHOULD
   NOT have a query or fragment component as any non-empty
   relative part in a reference would remove those parts from the
   resulting URI.

   In the absence of this parameter the scheme of the protocol,
   source address and source port of the registration request are
   assumed. That Base URI is constructed by concatenating the
   used protcol's scheme with the characters "://", the
   requester's source address as an address literal and ":"
   followed by its port (if it was not the protocol's default one)
   in analogy to [RFC7252] Section 6.5.

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

   If the registrant-ep uses an ephemeral port to register with,
   it MUST include the base parameter in the registration to
   provide a valid network path.

If the registrant-ep, located behind a NAT gateway, is
registering with a Resource Directory which is on the network
service side of the NAT gateway, the endpoint MUST use a
persistent port for the outgoing registration in order to
provide the NAT gateway with a valid network address for
replies and incoming requests.

Endpoints that register with a base that contains a path
component can not meaningfully use [RFC6690] Link Format due to
its prevalence of the Origin concept in relative reference
resolution; they can submit payloads for interpretation as
Modernized Link Format. Typically, links submitted by such an
endpoint are of the "path-noscheme" (starts with a path not
preceded by a slash, precisely defined in [RFC3986]
Section 3.3) form.

extra-attrs := Additional registration attributes (optional).
   The endpoint can pass any parameter registered at Section 10.3
   to the directory. If the RD is aware of the parameter's
   specified semantics, it processes it accordingly. Otherwise,
   it MUST store the unknown key and its value(s) as an endpoint
   attribute for further lookup.

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-Path option
   or Location header MUST be included in the response. This
   location MUST be a stable identifier generated by the RD as it is
   used for all subsequent operations on this registration resource.
   The registration resource location thus returned is for the
   purpose of updating the lifetime of the registration and for
   maintaining the content of the registered links, including
   updating and deleting links.

   A registration with an already registered ep and d value pair
   responds with the same success code and location as the original
   registration; the set of links registered with the endpoint is
   replaced with the links from the payload.

   The location MUST NOT have a query or fragment component, as that
   could conflict with query parameters during the Registration

   Update operation. Therefore, the Location-Query option MUST NOT
   be present in a successful response.

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

If the registration fails with a Service Unavailable response and a
Max-Age option or Retry-After header, the registering endpoint SHOULD
retry the operation after the time indicated. If the registration
fails in another way, including request timeouts, or if the Service
Unavailable error persists after several retries, or indicates a
longer time than the endpoint is willing to wait, it SHOULD pick
another registration URI from the "URI Discovery" step and if there
is only one or the list is exhausted, pick other choices from the
"Finding a Resource Directory" step. Care has to be taken to
consider the freshness of results obtained earlier, e.g. of the
result of a "/.well-known/core" response, the lifetime of an RDAO
option and of DNS responses. Any rate limits and persistent errors
from the "Finding a Resource Directory" step must be considered for
the whole registration time, not only for a single operation.

The following example shows a registrant-ep with the name "node1"
registering two resources to an RD using this interface. The
location "/rd" is an example RD location discovered in a request
similar to Figure 6.

Req: POST coap://rd.example.com/rd?ep=node1
Content-Format: 40
Payload:
</sensors/temp>;ct=41;rt="temperature-c";if="sensor";
      anchor="coap://spurious.example.com:5683",
</sensors/light>;ct=41;rt="light-lux";if="sensor"

Res: 2.01 Created
Location-Path: /rd/4521

                 Figure 7: Example registration payload

A Resource Directory may optionally support HTTP. Here is an example
of almost the same registration operation above, when done using HTTP
and the JSON Link Format.

```
Req: POST /rd?ep=node1&base=http://[2001:db8:1::1] HTTP/1.1
Host: example.com
Content-Type: application/link-format+json
Payload:
[
{"href": "/sensors/temp", "ct": "41", "rt": "temperature-c",
"if": "sensor", "anchor": "coap://spurious.example.com:5683"},
{"href": "/sensors/light", "ct": "41", "rt": "light-lux",
  "if": "sensor"}
]

Res: 201 Created
Location: /rd/4521
```

5.3.1. Simple Registration

Not all endpoints hosting resources are expected to know how to
upload links to an RD as described in Section 5.3. Instead, simple
endpoints can implement the Simple Registration approach described in
this section. An RD implementing this specification MUST implement
Simple Registration. However, there may be security reasons why this
form of directory discovery would be disabled.

This approach requires that the registrant-ep makes available the
hosted resources that it wants to be discovered, as links on its
"/.well-known/core" interface as specified in [RFC6690]. The links
in that document are subject to the same limitations as the payload
of a registration (with respect to Appendix D).

The registrant-ep finds one or more addresses of the directory server
as described in Section 4.

The registrant-ep asks the selected directory server to probe its
/.well-known/core and publish the links as follows:

The registrant-ep sends (and regularly refreshes with) a POST request
to the "/.well-known/core" URI of the directory server of choice.
The body of the POST request is empty, and triggers the resource
directory server to perform GET requests at the requesting
registrant-ep's /.well-known/core to obtain the link-format payload
to register.

The registrant-ep includes the same registration parameters in the
POST request as it would per Section 5.3. The registration base URI
of the registration is taken from the requesting server's URI.

The Resource Directory MUST NOT query the registrant-ep's data before
sending the response; this is to accommodate very limited endpoints.

The success condition only indicates that the request was valid (i.e.
the passed parameters are valid per se), not that the link data could
be obtained or parsed or was successfully registered into the RD.

The simple registration request interface is specified as follows:

Interaction: EP -> RD

Method: POST

URI Template: /.well-known/core{?ep,d,lt,extra-attrs*}

URI Template Variables are as they are for registration in
Section 5.3. The base attribute is not accepted to keep the
registration interface simple; that rules out registration over CoAP-
over-TCP or HTTP that would need to specify one.

The following response codes are defined for this interface:

Success: 2.04 "Changed".

Failure: 4.00 "Bad Request". Malformed request.

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

HTTP support: NO

For the second interaction triggered by the above, the registrant-ep
takes the role of server and the RD the role of client. (Note that
this is exactly the Well-Known Interface of [RFC6690] Section 4):

Interaction: RD -> EP

Method: GET

URI Template: /.well-known/core

The following response codes are defined for this interface:

Success: 2.05 "Content".

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.04 "Not Found". /.well-known/core does not exist or is
   empty.

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

   HTTP support: NO

   The registration resources MUST be deleted after the expiration of
   their lifetime. Additional operations on the registration resource
   cannot be executed because no registration location is returned.

   The following example shows a registrant-ep using Simple
   Registration, by simply sending an empty POST to a resource
   directory.

   Req:(to RD server from [2001:db8:2::1])
   POST /.well-known/core?lt=6000&ep=node1
   No payload

   Res: 2.04 Changed

   (later)

   Req: (from RD server to [2001:db8:2::1])
   GET /.well-known/core
   Accept: 40

   Res: 2.05 Content
   Content-Format: 40
   Payload:
   </sen/temp>

5.3.2. Third-party registration

   For some applications, even Simple Registration may be too taxing for
   some 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 party device, called a
   Commissioning Tool (CT). The commissioning tool can fill the
   Resource Directory from a database or other means. For that purpose
   scheme, IP address and port of the URI of the registered device is
   the value of the "base" parameter of the registration described in
   Section 5.3.

   It should be noted that the value of the "base" parameter applies to
   all the links of the registration and has consequences for the anchor
   value of the individual links as exemplified in Appendix B. An

   eventual (currently non-existing) "base" attribute of the link is not
   affected by the value of "base" parameter in the registration.

6. RD Groups

   This section defines the REST API for the creation, management, and
   lookup of endpoints for group operations. Similar to endpoint
   registration entries in the RD, groups 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 commissioning tool (CT) used to
   configure groups, makes a request to the RD indicating the name of
   the group to create (or update), optionally the sector the group
   belongs to, and optionally the multicast address of the group. This
   specification does not require that the endpoints belong to the same
   sector as the group, but a Resource Directory implementation can
   impose requirements on the sectors of groups and endpoints depending
   on its configuration.

   The registration message is a list of links to registration resources
   of the endpoints that belong to that group. The CT can use any URI
   reference discovered using endpoint lookup from the same server or
   obtained by registering an endpoint using third party registration
   and enter it into a group.

   The commissioning tool SHOULD not send any target attributes with the
   links to the registration resources, and the resource directory
   SHOULD reject registrations that contain links with unprocessable
   attributes.

   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: CT -> RD

   Method: POST

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

   URI Template Variables:

rd-group := RD Group URI (mandatory). This is the location of
   the RD Group REST API.

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

d := Sector (optional). The sector to which this group belongs.
   The maximum length of this parameter is 63 bytes. When this
   parameter is not present, the RD MAY associate the group with a
   configured default sector or leave it empty.

base := Group Base URI (optional). This parameter sets the
   scheme, address and port of the multicast address associated
   with the group. When base is used, scheme and host are
   mandatory and port parameter is optional.

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 or
   Location-Path option MUST be returned in response to a successful
   group CREATE operation. This location MUST be a stable identifier
   generated by the RD as it is used for delete operations of the
   group resource.

   As with the Registration operation, the location MUST NOT have a
   query or fragment component.

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. The RD group path /rd-group is an
example RD location discovered in a request similar to Figure 6.

```
   Req: POST coap://rd.example.com/rd-group?gp=lights
                                  &base=coap://[ff35:30:2001:db8::1]
   Content-Format: 40
   Payload:
   </rd/4521>,
   </rd/4522>

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

   A relative href value denotes the path to the registration resource
   of the Endpoint. When pointing to a registration resource on a
   different RD, the href value is a URI.

6.2. Group Removal

   A group can be removed simply by sending a removal message to the
   location of the group registration resource which was returned when
   initially 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: CT -> RD

   Method: DELETE

   URI Template: {+location}

   URI Template Variables:

      location := This is the path of the group resource 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 with the example location value /rd-group/12.

Req: DELETE /rd-group/12

Res: 2.02 Deleted

7. RD Lookup

To discover the resources registered with the RD, a lookup interface
must be provided. 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).

RD Lookup allows lookups for groups, endpoints and resources using
attributes defined in this document 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. Thus, a group lookup MUST
return a list of groups, an endpoint lookup MUST return a list of
endpoints and a resource lookup MUST return a list of links to
resources.

The lookup type is selected by a URI endpoint, which is indicated by
a Resource Type as per Table 1 below:

```
+-------------+-------------------+-----------+
| Lookup Type | Resource Type     | Mandatory |
+-------------+-------------------+-----------+
| Resource    | core.rd-lookup-res | Mandatory |
| Endpoint    | core.rd-lookup-ep  | Mandatory |
| Group       | core.rd-lookup-gp  | Optional  |
+-------------+-------------------+-----------+
```

Table 1: Lookup Types

7.1. Resource lookup

Resource lookup results in links that are semantically equivalent to
the links submitted to the RD. The links and link parameters
returned by the lookup are equal to the submitted ones, except that
the target and anchor references are fully resolved.

Links that did not have an anchor attribute are therefore returned
with the base URI of the registration as the anchor. Links of which
href or anchor was submitted as a (full) URI are returned with these
attributes unmodified.

Above rules allow the client to interpret the response as links
without any further knowledge of the storage conventions of the RD.
The Resource Directory MAY replace the registration base URIs with a
configured intermediate proxy, e.g. in the case of an HTTP lookup
interface for CoAP endpoints.

7.2. Lookup filtering

Using the Accept Option, the requester can control whether the
returned 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 page and count parameters are used to obtain lookup results in
specified increments using pagination, where count specifies how many
links to return and page specifies which subset of links organized in
sequential pages, each containing 'count' links, starting with link
zero and page zero. Thus, specifying count of 10 and page of 0 will
return the first 10 links in the result set (links 0-9). Count = 10
and page = 1 will return the next 'page' containing links 10-19, and
so on.

Multiple search criteria MAY be included in a lookup. All included
criteria MUST match for a link to be returned. The Resource
Directory MUST support matching with multiple search criteria.

A link matches a search criterion if it has an attribute of the same
name and the same value, allowing for a trailing "*" wildcard
operator as in Section 4.1 of [RFC6690]. Attributes that are defined
as "link-type" match if the search value matches any of their values
(see Section 4.1 of [RFC6690]; e.g. "?if=core.s" matches ";if="abc
core.s";"). A link also matches a search criterion if the link that
would be produced for any of its containing entities would match the
criterion, or an entity contained in it would: A search criterion
matches an endpoint if it matches the endpoint itself, any of the
groups it is contained in or any resource it contains. A search
criterion matches a resource if it matches the resource itself, the
resource's endpoint, or any of the endpoint's groups.

Note that "href" is a valid search criterion and matches target
references. Like all search criteria, on a resource lookup it can
match the target reference of the resource link itself, but also the
registration resource of the endpoint that registered it, or any
group resource that endpoint is contained in. Queries for resource
link targets MUST be in URI form (i.e. not relative references) and
are matched against a resolved link target. Queries for groups and
endpoints SHOULD be expressed in path-absolute form if possible and

MUST be expressed in URI form otherwise; the RD SHOULD recognize
either.

Endpoints that are interested in a lookup result repeatedly or
continuously can use mechanisms like ETag caching, resource
observation ([RFC7641]), or any future mechanism that might allow
more efficient observations of collections. These are advertised,
detected and used according to their own specifications and can be
used with the lookup interface as with any other resource.

When resource observation is used, every time the set of matching
links changes, or the content of a matching link changes, the RD
sends a notification with the matching link set. The notification
contains the successful current response to the given request,
especially with respect to representing zero matching links (see
"Success" item below).

The lookup interface is specified as follows:

Interaction: Client -> RD

Method: GET

URI Template: {+type-lookup-location}{?page,count,search*}

URI Template Variables:

   type-lookup-location := RD Lookup URI for a given lookup type
      (mandatory). The address is discovered as described in
      Section 5.2.

   search := Search criteria for limiting the number of results
      (optional).

   page := Page (optional). Parameter cannot be used without the
      count parameter. Results are returned from result set in pages
      that contain 'count' links starting from index (page * count).
      Page numbering starts with zero.

   count := Count (optional). Number of results is limited to this
      parameter value. If the page parameter is also present, the
      response MUST only include 'count' links starting with the
      (page * count) link in the result set from the query. If the
      count parameter is not present, then the response MUST return
      all matching links in the result set. Link numbering starts
      with zero.

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

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

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

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.
   The payload can contain zero links (which is an empty payload,
   "80" (hex) or "[]" in the respective content format), indicating
   that no entities matched the 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 group and endpoint lookup return registration resources which can
only be manipulated by the registering endpoint. Examples of group
and endpoint lookup belong to the management aspects of the RD and
are shown in Appendix A.5. The resource lookup examples are shown in
this section.

7.3. Resource lookup examples

The examples in this section assume the existence of CoAP hosts with
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
with the example resource look-up locations discovered in Figure 6:

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

Res: 2.05 Content
<coap://[2001:db8:3::123]:61616/temp>;rt="temperature";
         anchor="coap://[2001:db8:3::123]:61616"

The same lookup using the CBOR Link Format media type:

```
Req: GET /rd-lookup/res?rt=temperature
Accept: TBD64

Res: 2.05 Content
Content-Format: TBD64
Payload in Hex notation:
81A3017823636F61703A2F2F5B323030313A6462383A333A3A3132335D3A363136313
62F74656D7003781E636F61703A2F2F5B323030313A6462383A333A3A3132335D3A36
31363136096B74656D7065726174757265
Decoded payload:
[{1: "coap://[2001:db8:3::123]:61616/temp", 9: "temperature",
3: "coap://[2001:db8:3::123]:61616"}]
```

A client that wants to be notified of new resources as they show up
can use observation:

```
Req: GET /rd-lookup/res?rt=light
Observe: 0

Res: 2.05 Content
Observe: 23
Payload: empty

(at a later point in time)

Res: 2.05 Content
Observe: 24
Payload:
<coap://[2001:db8:3::124]/west>;rt="light";
    anchor="coap://[2001:db8:3::124]",
<coap://[2001:db8:3::124]/south>;rt="light";
    anchor="coap://[2001:db8:3::124]",
<coap://[2001:db8:3::124]/east>;rt="light";
    anchor="coap://[2001:db8:3::124]"
```

The following example shows a client performing a paginated resource
lookup

```
Req: GET /rd-lookup/res?page=0&count=5

Res: 2.05 Content
<coap://[2001:db8:3::123]:61616/res/0>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/1>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/2>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/3>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/4>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616"

Req: GET /rd-lookup/res?page=1&count=5

Res: 2.05 Content
<coap://[2001:db8:3::123]:61616/res/5>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/6>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/7>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/8>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616",
<coap://[2001:db8:3::123]:61616/res/9>;rt=sensor;ct=60;
    anchor="coap://[2001:db8:3::123]:61616"
```

The following example shows a client performing a lookup of all
resources from endpoints of all endpoints of a given endpoint type.
It assumes that two endpoints (with endpoint names "sensor1" and
"sensor2") have previously registered with their respective addresses
"coap://sensor1.example.com" and "coap://sensor2.example.com", and
posted the very payload of the 6th request of section 5 of [RFC6690].

It demonstrates how absolute link targets stay unmodified, while
relative ones are resolved:

```
Req: GET /rd-lookup/res?et=oic.d.sensor

<coap://sensor1.example.com/sensors>;ct=40;title="Sensor Index";
    anchor="coap://sensor1.example.com",
<coap://sensor1.example.com/sensors/temp>;rt="temperature-c";
    if="sensor"; anchor="coap://sensor1.example.com",
<coap://sensor1.example.com/sensors/light>;rt="light-lux";
    if="sensor"; anchor="coap://sensor1.example.com",
<http://www.example.com/sensors/t123>;rel="describedby";
    anchor="coap://sensor1.example.com/sensors/temp",
<coap://sensor1.example.com/t>;rel="alternate";
    anchor="coap://sensor1.example.com/sensors/temp",
<coap://sensor2.example.com/sensors>;ct=40;title="Sensor Index";
    anchor="coap://sensor2.example.com",
<coap://sensor2.example.com/sensors/temp>;rt="temperature-c";
    if="sensor"; anchor="coap://sensor2.example.com",
<coap://sensor2.example.com/sensors/light>;rt="light-lux";
    if="sensor"; anchor="coap://sensor2.example.com",
<http://www.example.com/sensors/t123>;rel="describedby";
    anchor="coap://sensor2.example.com/sensors/temp",
<coap://sensor2.example.com/t>;rel="alternate";
    anchor="coap://sensor2.example.com/sensors/temp"
```

8. Security policies

The Resource Directory (RD) provides assistance to applications
situated on a selection of nodes to discover endpoints on connected
nodes. This section discusses different security aspects of
accessing the RD.

The contents of the RD are inserted in two ways:

1. The node hosting the discoverable endpoint fills the RD with the
    contents of /.well-known/core by:

    * Storing the contents directly into RD (see Section 5.3)

    * Requesting the RD to load the contents from /.well-known/core
       see (section {{simple}})

2. A Commissioning Tool (CT) fills the RD with endpoint information
    for a set of discoverable nodes. (see Section 5.3 with
    base=authority parameter value)

In both cases, the nodes filling the RD should be authenticated and
authorized to change the contents of the RD. An Authorization Server
(AS) is responsible to assign a token to the registering node to

authorize the node to discover or register endpoints in a given RD
[I-D.ietf-ace-oauth-authz].

It can be imagined that an installation is divided in a set of
security regions, each one with its own RD(s) to discover the
endpoints that are part of a given security region. An endpoint that
wants to discover an RD, responsible for a given region, needs to be
authorized to learn the contents of a given RD. Within a region, for
a given RD, a more fine-grained security division is possible based
on the values of the endpoint registration parameters. Authorization
to discover endpoints with a given set of filter values is
recommended for those cases.

When a node registers its endpoints, criteria are needed to authorize
the node to enter them. An important aspect is the uniqueness of the
(endpoint name, and optional sector) pair within the RD. Consider
the two cases separately: (1) CT registers endpoints, and (2) the
registering node registers its own endpoint(s). * A CT needs
authorization to register a set of endpoints. This authorization can
be based on the region, i.e. a given CT is authorized to register any
endpoint (endpoint name, sector) into a given RD, or to register an
endpoint with (endpoint name, sector) value pairs assigned by the AS,
or can be more fine-grained, including a subset of registration
parameter values. * A given endpoint that registers itself, needs to
proof its possession of its unique (endpoint name, sector) value
pair. Alternatively, the AS can authorize the endpoint to register
with an (endpoint name, sector) value pair assigned by the AS. * A
separate document needs to specify these aspects to ensure
interoperability between registering nodes and RD. The subsections
below give some hints how to handle a subset of the different
aspects.

8.1. Secure RD discovery

The Resource Server (RS) discussed in [I-D.ietf-ace-oauth-authz] is
equated to the RD. The client (C) needs to discover the RD as
discussed in Section 4. C can discover the related AS by sending a
request to the RD. The RD denies the request by sending the address
of the related AS, as discussed in section 5.1 of
[I-D.ietf-ace-oauth-authz]. The client MUST send an authorization
request to the AS. When appropriate, the AS returns a token that
specifies the authorization permission which needs to be specified in
a separate document.

8.2. Secure RD filtering

   The authorized parameter values for the queries by a given endpoint
   must be registered by the AS. The AS communicates the parameter
   values in the token. A separate document needs to specify the
   parameter value combinations and their storage in the token. The RD
   decodes the token and checks the validity of the queries of the
   client.

8.3. Secure endpoint Name assignment

   This section only considers the assignment of a name to the endpoint
   based on an automatic mechanism without use of AS. More elaborate
   protocols are out of scope. The registering endpoint is authorized
   by the AS to discover the RD and add registrations. A token is
   provided by the AS and communicated from registering endpoint to RD.
   It is assumed that DTLS is used to secure the channel between
   registering endpoint and RD, where the registering endpoint is the
   DTLS client. Assuming that the client is provided by a certificate
   at manufacturing time, the certificate is uniquely identified by the
   CN field and the serial number. The RD can assign a unique endpoint
   name by using the certificate identifier as endpoint name. Proof of
   possession of the endpoint name by the registering endpoint is
   checked by encrypting the certificate identifier with the private key
   of the registering endpoint, which the RD can decrypt with the public
   key stored in the certificate. Even simpler, the authorized
   registering endpoint can generate a random number (or string) that
   identifies the endpoint. The RD can check for the improbable
   replication of the random value. The RD MUST check that registering
   endpoint uses only one random value for each authorized endpoint.

9. 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.

9.1. Endpoint Identification and Authentication

   An Endpoint (name, sector) pair is unique within the et of endpoints
   regsitered by the RD. 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 on a resource directory
SHOULD be mutually authenticated using Pre-Shared Key, Raw Public Key
or Certificate based security.

Consider the following threat: two devices A and B are registered at
a single server. Both devices have unique, per-device credentials
for use with DTLS to make sure that only parties with authorization
to access A or B can do so.

Now, imagine that a malicious device A wants to sabotage the device
B. It uses its credentials during the DTLS exchange. Then, it
specifies the endpoint name of device B as the name of its own
endpoint in device A. If the server does not check whether the
identifier provided in the DTLS handshake matches the identifier used
at the CoAP layer then it may be inclined to use the endpoint name
for looking up what information to provision to the malicious device.

Section 8.3 specifies an example that removes this threat for
endpoints that have a certificate installed.

9.2. Access Control

Access control SHOULD be performed separately for the RD
registration, Lookup, and group API paths, 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 sector, endpoint or resource
level.

9.3. 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. There is also a danger that NTP Servers could become
implicated in denial-of-service (DoS) attacks since they run on
unprotected UDP, there is no return routability check, and they 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.

10. IANA Considerations

10.1. Resource Types

IANA is asked to enter the following values into the Resource Type
(rt=) Link Target Attribute Values sub-registry of the Constrained
Restful Environments (CoRE) Parameters registry defined in [RFC6690]:

```
+--------------------+----------------------------+-----------------+
| Value              | Description                | Reference       |
+--------------------+----------------------------+-----------------+
| core.rd            | Directory resource of an   | RFCTHIS Section |
|                    | RD                         | 5.2             |
| core.rd-group      | Group directory resource   | RFCTHIS Section |
|                    | of an RD                   | 5.2             |
| core.rd-lookup-res | Resource lookup of an RD   | RFCTHIS Section |
|                    |                            | 5.2             |
| core.rd-lookup-ep  | Endpoint lookup of an RD   | RFCTHIS Section |
|                    |                            | 5.2             |
| core.rd-lookup-gp  | Group lookup of an RD      | RFCTHIS Section |
|                    |                            | 5.2             |
| core.rd-ep         | Endpoint resource of an RD | RFCTHIS Section |
|                    |                            | 7               |
| core.rd-gp         | Group resource of an RD    | RFCTHIS Section |
|                    |                            | 7               |
+--------------------+----------------------------+-----------------+
```

10.2. IPv6 ND Resource Directory Address Option

This document registers one new ND option type under the sub-registry
"IPv6 Neighbor Discovery Option Formats":

o Resource Directory address Option (38)

10.3. 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

o the human readable name of the parameter,

o the short name as used in query parameters or link attributes,

o indication of whether it can be passed as a query parameter at
   registration of endpoints or groups, as a query parameter in
   lookups, or be expressed as a link attribute,

o validity requirements if any, and

o a description.

The query parameter MUST be both a valid URI query key [RFC3986] and
a parmname as used in [RFC5988].

The description must give details on which registrations they apply
to (Endpoint, group registrations or both? Can they be updated?),
and how they are to be processed in lookups.

The mechanisms around new RD parameters should be designed in such a
way that they tolerate RD implementations that are unaware of the
parameter and expose any parameter passed at registration or updates
on in endpoint lookups. (For example, if a parameter used at
registration were to be confidential, the registering endpoint should
be instructed to only set that parameter if the RD advertises support
for keeping it confidential at the discovery step.)

Initial entries in this sub-registry are as follows:

```
+--------------+-------+--------------+-----+---------------------+
| Full name    | Short | Validity     | Use | Description         |
+--------------+-------+--------------+-----+---------------------+
| Endpoint     | ep    |              | RLA | Name of the         |
| Name         |       |              |     | endpoint, max 63    |
|              |       |              |     | bytes               |
| Lifetime     | lt    | 60-4294967295| R   | Lifetime of the     |
|              |       |              |     | registration in     |
|              |       |              |     | seconds             |
| Sector       | d     |              | RLA | Sector to which this|
|              |       |              |     | endpoint belongs    |
| Registration | base  | URI          | RLA | The scheme, address |
| Base URI     |       |              |     | and port and path at|
|              |       |              |     | which this server is|
|              |       |              |     | available           |
| Group Name   | gp    |              | RLA | Name of a group in  |
|              |       |              |     | the RD              |
| Page         | page  | Integer      | L   | Used for pagination |
| Count        | count | Integer      | L   | Used for pagination |
| Endpoint     | et    |              | RLA | Semantic name of the|
| Type         |       |              |     | endpoint (see       |
|              |       |              |     | Section 10.4)       |
+--------------+-------+--------------+-----+---------------------+
```

                        Table 2: RD Parameters

(Short: Short name used in query parameters or link attributes. Use:
R = used at registration, L = used at lookup, A = expressed in link
attribute

The descriptions for the options defined in this document are only
summarized here. To which registrations they apply and when they are
to be shown is described in the respective sections of this document.

The IANA policy for future additions to the sub-registry is "Expert
Review" as described in [RFC8126]. The evaluation should consider
formal criteria, duplication of functionality (Is the new entry
redundant with an existing one?), topical suitability (E.g. is the
described property actually a property of the endpoint and not a
property of a particular resource, in which case it should go into
the payload of the registration and need not be registered?), and the
potential for conflict with commonly used link attributes (For
example, "if" could be used as a parameter for conditional
registration if it were not to be used in lookup or attributes, but
would make a bad parameter for lookup, because a resource lookup with
an "if" query parameter could ambiguously filter by the registered
endpoint property or the [RFC6690] link attribute). It is expected

   that the registry will receive between 5 and 50 registrations in
   total over the next years.

10.3.1. Full description of the "Endpoint Type" Registration Parameter

   An endpoint registering at an RD can describe itself with endpoint
   types, similar to how resources are described with Resource Types in
   [RFC6690]. An endpoint type is expressed as a string, which can be
   either a URI or one of the values defined in the Endpoint Type sub-
   registry. Endpoint types can be passed in the "et" query parameter
   as part of extra-attrs at the Registration step, are shown on
   endpoint lookups using the "et" target attribute, and can be filtered
   for using "et" as a search criterion in resource and endpoint lookup.
   Multiple endpoint types are given as separate query parameters or
   link attributes.

   Note that Endpoint Type differs from Resource Type in that it uses
   multiple attributes rather than space separated values. As a result,
   Resource Directory implementations automatically support correct
   filtering in the lookup interfaces from the rules for unknown
   endpoint attributes.

10.4. "Endpoint Type" (et=) RD Parameter values

   This specification establishes a new sub-registry under "CoRE
   Parameters" called '"Endpoint Type" (et=) RD Parameter values'. The
   registry properties (required policy, requirements, template) are
   identical to those of the Resource Type parameters in [RFC6690], in
   short:

   The review policy is IETF Review for values starting with "core", and
   Specification Required for others.

   The requirements to be enforced are:

   o The values MUST be related to the purpose described in
      Section 10.3.1.

   o The registered values MUST conform to the ABNF reg-rel-type
      definition of [RFC6690] and MUST NOT be a URI.

   o It is recommended to use the period "." character for
      segmentation.

   The registry is initially empty.

10.5. **Multicast Address Registration**

   IANA has assigned the following multicast addresses for use by CoAP
   nodes:

   IPv4 - "all CoRE resource directories" address, from the "IPv4
   Multicast Address Space Registry" equal to "All CoAP Nodes",
   224.0.1.187. As the address is used for discovery that may span
   beyond a single network, it has come from the Internetwork Control
   Block (224.0.1.x, RFC 5771).

   IPv6 - "all CoRE resource directories" address MCD1 (suggestions
   FF0X::FE), from the "IPv6 Multicast Address Space Registry", in the
   "Variable Scope Multicast Addresses" space (RFC 3307). Note that
   there is a distinct multicast address for each scope that interested
   CoAP nodes should listen to; CoAP needs the Link-Local and Site-Local
   scopes only.

11. **Examples**

   Two examples are presented: a Lighting Installation example in
   Section 11.1 and a LWM2M example in Section 11.2.

11.1. **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
   or naming procedures, because the example only "emphasizes" some of
   the issues that may influence the use of the RD and does not pretend
   to be normative.

11.1.1. **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 endpoint. 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.

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          | 2001:db8:4::1  |
       | luminary2          | 2001:db8:4::2  |
       | Presence sensor    | 2001:db8:4::3  |
       | Resource directory | 2001:db8:4::ff |
       +--------------------+----------------+
```

               Table 3: interface SLAAC addresses

In Section 11.1.2 the use of resource directory during installation
is presented.

11.1.2. 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 endpoints 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 endpoint 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           | endpoint         | 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

It is assumed that the CT knows the RD's address, and has performed
URI discovery on it that returned a response like the one in the
Section 5.2 example.

The CT inserts the endpoints of the luminaries and the sensor in the
RD using the registration base URI parameter (base) to specify the
interface address:

```
Req: POST coap://[2001:db8:4::ff]/rd
  ?ep=lm_R2-4-015_wndw&base=coap://[2001:db8:4::1]&d=R2-4-015
Payload:
</light/left>;rt="light",
</light/middle>;rt="light",
</light/right>;rt="light"

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

```
Req: POST coap://[2001:db8:4::ff]/rd
  ?ep=lm_R2-4-015_door&base=coap://[2001:db8:4::2]&d=R2-4-015
Payload:
</light/left>;rt="light",
</light/middle>;rt="light",
</light/right>;rt="light"

Res: 2.01 Created
Location-Path: /rd/4522

Req: POST coap://[2001:db8:4::ff]/rd
  ?ep=ps_R2-4-015_door&base=coap://[2001:db8:4::3]d&d=R2-4-015
Payload:
</ps>;rt="p-sensor"

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

The sector name d=R2-4-015 has been added for an efficient lookup
because filtering on "ep" name is more awkward. The same sector name
is communicated to the two luminaries and the presence sensor by the
CT.

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

```
Req: POST coap://[2001:db8:4::ff]/rd-group
?gp=grp_R2-4-015&base=coap://[ff05::1]
Payload:
</rd/4521>,
</rd/4522>,
</rd/4523>

Res: 2.01 Created
Location-Path: /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 sector and own IPv6 address, looks up the
groups containing light resources it is assigned to:

```
Req: GET coap://[2001:db8:4::ff]/rd-lookup/gp
  ?d=R2-4-015&base=coap://[2001:db8:4::1]&rt=light

Res: 2.05 Content
</rd-group/501>;gp="grp_R2-4-015";base="coap://[ff05::1]"
```

From the returned base 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 coap://[2001:db8:4::1]/coap-group
Content-Format: application/coap-group+json
Payload:
{ "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.

11.2. 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(OMA Name
Authority). 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 sectors 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, and applications are currently out of scope for
LWM2M.

The location of the LWM2M Server and RD URI path is provided by the
LWM2M Bootstrap process, so no dynamic discovery of the RD is used.
LWM2M Servers and endpoints are not required to implement the /.well-
known/core resource.

11.2.1. 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".

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

object-id := OMNA (OMA Name Authority) 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 (OMA Name Authority) 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).

11.2.2. LWM2M Register Endpoint

LWM2M defines a registration interface based on the REST API,
described in Section 5. The RD registration URI path of the LWM2M
Resource Directory is specified to be "/rd".

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 its resources are available for
management and service enablement (REST API) operations.

LWM2M endpoints may use the following RD registration parameters as
defined in Table 2 :

ep - Endpoint Name
lt - registration lifetime

Endpoint Name, Lifetime, and LWM2M Version are mandatory parameters
for the register operation, all other registration parameters are
optional.

Additional optional LWM2M registration parameters are defined:

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

             Table 5: LWM2M Additional Registration Parameters

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

et - Endpoint Type
base - Registration Base URI

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.

11.2.3. LWM2M Update Endpoint Registration

The LwM2M update is really very similar to the registration update as
described in Appendix A.1, with the only difference that there are
more parameters defined and available. All the parameters listed in
that section are also available with the initial registration but are
all optional:

      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.

11.2.4. 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 Appendix A.2.

12. Acknowledgments

      Oscar Novo, Srdjan Krco, Szymon Sasin, Kerry Lynn, Esko Dijk, Anders
      Brandt, Matthieu Vial, Jim Schaad, Mohit Sethi, Hauke Petersen,
      Hannes Tschofenig, Sampo Ukkola, Linyi Tian, and Jan Newmarch have
      provided helpful comments, discussions and ideas to improve and shape
      this document. Zach would also like to thank his colleagues from the
      EU FP7 SENSEI project, where many of the resource directory concepts
      were originally developed.

13. Changelog

      changes from -14 to -15

      o Rewrite of section "Security policies"

      o Clarify that the "base" parameter text applies both to relative
         references both in anchor and href

      o Renamed "Registree-EP" to Registrant-EP"

      o Talk of "relative references" and "URIs" rather than "relative"
         and "absolute" URIs. (The concept of "absolute URIs" of [RFC3986]
         is not needed in RD).

      o Fixed examples

      o Editorial changes

      changes from -13 to -14

o Rename "registration context" to "registration base URI" (and
   "con" to "base") and "domain" to "sector" (where the abbreviation
   "d" stays for compatibility reasons)

o Introduced resource types core.rd-ep and core.rd-gp

o Registration management moved to appendix A, including endpoint
   and group lookup

o Minor editorial changes

   * PATCH/iPATCH is clearly deferred to another document

   * Recommend against query / fragment identifier in con=

   * Interface description lists are described as illustrative

   * Rewording of Simple Registration

o Simple registration carries no error information and succeeds
   immediately (previously, sequence was unspecified)

o Lookup: href are matched against resolved values (previously, this
   was unspecified)

o Lookup: lt are not exposed any more

o con/base: Paths are allowed

o Registration resource locations can not have query or fragment
   parts

o Default life time extended to 25 hours

o clarified registration update rules

o lt-value semantics for lookup clarified.

o added template for simple registration

changes from -12 to -13

o Added "all resource directory" nodes MC address

o Clarified observation behavior

o version identification

   o example rt= and et= values

   o domain from figure 2

   o more explanatory text

   o endpoints of a groups hosted by different RD

   o resolve RFC6690-vs-8288 resolution ambiguities:

      * require registered links not to be relative when using anchor

      * return absolute URIs in resource lookup

   changes from -11 to -12

   o added Content Model section, including ER diagram

   o removed domain lookup interface; domains are now plain attributes
      of groups and endpoints

   o updated chapter "Finding a Resource Directory"; now distinguishes
      configuration-provided, network-provided and heuristic sources

   o improved text on: atomicity, idempotency, lookup with multiple
      parameters, endpoint removal, simple registration

   o updated LWM2M description

   o clarified where relative references are resolved, and how context
      and anchor interact

   o new appendix on the interaction with RFCs 6690, 5988 and 3986

   o lookup interface: group and endpoint lookup return group and
      registration resources as link targets

   o lookup interface: search parameters work the same across all
      entities

   o removed all methods that modify links in an existing registration
      (POST with payload, PATCH and iPATCH)

   o removed plurality definition (was only needed for link
      modification)

   o enhanced IANA registry text

   o   state that lookup resources can be observable

   o   More examples and improved text

   changes from -09 to -10

   o   removed "ins" and "exp" link-format extensions.

   o   removed all text concerning DNS-SD.

   o   removed inconsistency in RDAO text.

   o   suggestions taken over from various sources

   o   replaced "Function Set" with "REST API", "base URI", "base path"

   o   moved simple registration to registration section

   changes from -08 to -09

   o   clarified the "example use" of the base RD resource values /rd,
       /rd-lookup, and /rd-group.

   o   changed "ins" ABNF notation.

   o   various editorial improvements, including in examples

   o   clarifications for RDAO

   changes from -07 to -08

   o   removed link target value returned from domain and group lookup
       types

   o   Maximum length of domain parameter 63 bytes for consistency with
       group

   o   removed option for simple POST of link data, don't require a
       .well-known/core resource to accept POST data and handle it in a
       special way; we already have /rd for that

   o   add IPv6 ND Option for discovery of an RD

   o   clarify group configuration section 6.1 that endpoints must be
       registered before including them in a group

   o   removed all superfluous client-server diagrams

o   simplified lighting example

o   introduced Commissioning Tool

o   RD-Look-up text is extended.

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](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.

14. References

14.1. Normative References

[I-D.ietf-core-links-json]
          Li, K., Rahman, A., and C. Bormann, "Representing
          Constrained RESTful Environments (CoRE) Link Format in
          JSON and CBOR", draft-ietf-core-links-json-10 (work in
          progress), February 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <https://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,
          <https://www.rfc-editor.org/info/rfc3986>.

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

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

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

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
          Writing an IANA Considerations Section in RFCs", BCP 26,
          RFC 8126, DOI 10.17487/RFC8126, June 2017,
          <https://www.rfc-editor.org/info/rfc8126>.

14.2. Informative References

[ER]   Chen, P., "The entity-relationship model---toward a
          unified view of data", ACM Transactions on Database
          Systems Vol. 1, pp. 9-36, DOI 10.1145/320434.320440, March
          1976.

   [I-D.arkko-core-dev-urn]
              Arkko, J., Jennings, C., and Z. Shelby, "Uniform Resource
              Names for Device Identifiers", draft-arkko-core-dev-urn-05
              (work in progress), October 2017.

   [I-D.bormann-t2trg-rel-impl]
              Bormann, C., "impl-info: A link relation type for
              disclosing implementation information", draft-bormann-
              t2trg-rel-impl-00 (work in progress), January 2018.

   [I-D.ietf-ace-oauth-authz]
              Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and
              H. Tschofenig, "Authentication and Authorization for
              Constrained Environments (ACE) using the OAuth 2.0
              Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-16
              (work in progress), October 2018.

   [I-D.ietf-anima-bootstrapping-keyinfra]
              Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
              S., and K. Watsen, "Bootstrapping Remote Secure Key
              Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
              keyinfra-16 (work in progress), June 2018.

   [I-D.silverajan-core-coap-protocol-negotiation]
              Silverajan, B. and M. Ocak, "CoAP Protocol Negotiation",
              draft-silverajan-core-coap-protocol-negotiation-09 (work
              in progress), July 2018.

   [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
              Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
              Transfer Protocol -- HTTP/1.1", RFC 2616,
              DOI 10.17487/RFC2616, June 1999,
              <https://www.rfc-editor.org/info/rfc2616>.

   [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,
              <https://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,
              <https://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,
              <https://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,
              <https://www.rfc-editor.org/info/rfc7390>.

   [RFC7641] Hartke, K., "Observing Resources in the Constrained
              Application Protocol (CoAP)", RFC 7641,
              DOI 10.17487/RFC7641, September 2015,
              <https://www.rfc-editor.org/info/rfc7641>.

   [RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and
              FETCH Methods for the Constrained Application Protocol
              (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017,
              <https://www.rfc-editor.org/info/rfc8132>.

   [RFC8288] Nottingham, M., "Web Linking", RFC 8288,
              DOI 10.17487/RFC8288, October 2017,
              <https://www.rfc-editor.org/info/rfc8288>.

   [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig,
              "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392,
              May 2018, <https://www.rfc-editor.org/info/rfc8392>.

Appendix A. Registration Management

   This section describes how the registering endpoint can maintain the
   registries that it created. The registering endpoint can be the
   registrant-ep or the CT. An endpoint SHOULD NOT use this interface
   for registries that it did not create. The registries are resources
   of the RD.

   After the initial registration, the registering endpoint retains the
   returned location of the Registration Resource for further
   operations, including refreshing the registration in order to extend
   the lifetime and "keep-alive" the registration. When the lifetime of
   the registration has expired, the RD SHOULD NOT respond to discovery
   queries concerning this endpoint. The RD SHOULD continue to provide
   access to the Registration Resource after a registration time-out
   occurs in order to enable the registering endpoint to eventually
   refresh the registration. The RD MAY eventually remove the
   registration resource for the purpose of garbage collection and
   remove it from any group it belongs to. If the Registration Resource
   is removed, the corresponding endpoint will need to be re-registered.

The Registration Resource may also be used to inspect the
registration resource using GET, update the registration, cancel the
registration using DELETE, do an endpoint lookup, or a group lookup.

These operations are described below.

A.1. Registration Update

The update interface is used by the registering endpoint to refresh
or update its registration with an RD. To use the interface, the
registering endpoint sends a POST request to the registration
resource returned by the initial registration operation.

An update MAY update the lifetime- or the context- registration
parameters "lt", "base" as in Section 5.3. Parameters that are not
being changed SHOULD NOT be included in an update. Adding parameters
that have not changed increases the size of the message but does not
have any other implications. Parameters MUST be included as query
parameters in an update operation as in Section 5.3.

A registration update resets the timeout of the registration to the
(possibly updated) lifetime of the registration, independent of
whether a "lt" parameter was given.

If the context of the registration is changed in an update, relative
references submitted in the original registration or later updates
are resolved anew against the new context.

The registration update operation only describes the use of POST with
an empty payload. Future standards might describe the semantics of
using content formats and payloads with the POST method to update the
links of a registration (see Appendix A.4).

The update registration request interface is specified as follows:

Interaction: EP -> RD

Method: POST

URI Template: {+location}{?lt,con,extra-attrs*}

URI Template Variables:

   location := This is the Location 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,

the previous last lifetime set on a previous update or the
original registration (falling back to 90000) SHOULD be used.

base := Base URI (optional). This parameter updates the Base URI
   established in the original registration to a new value. If
   the parameter is set in an update, it is stored by the RD as
   the new Base URI under which to interpret the relative links
   present in the payload of the original registration, following
   the same restrictions as in the registration. If the parameter
   is not set in the request but was set before, the previous Base
   URI value is kept unmodified. If the parameter is not set in
   the request and was not set before either, the source address
   and source port of the update request are stored as the Base
   URI.

extra-attrs := Additional registration attributes (optional). As
   with the registration, the RD processes them if it knows their
   semantics. Otherwise, unknown attributes are stored as
   endpoint attributes, overriding any previously stored endpoint
   attributes of the same key.

Content-Format: none (no payload)

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

If the registration update fails with a "Service Unavailable"
response and a Max-Age option or Retry-After header, the registering
endpoint SHOULD retry the operation after the time indicated. If the
registration fails in another way, including request timeouts, or if
the time indicated exceeds the remaining lifetime, the registering
endpoint SHOULD attempt registration again.

   The following example shows how the registering endpoint updates its
   registration resource at an RD using this interface with the example
   location value: /rd/4521.

   Req: POST /rd/4521

   Res: 2.04 Changed

   The following example shows the registering endpoint updating its
   registration resource at an RD using this interface with the example
   location value: /rd/4521. The initial registration by the
   registering endpoint set the following values:

   o endpoint name (ep)=endpoint1

   o lifetime (lt)=500

   o Base URI (base)=coap://local-proxy-old.example.com:5683

   o payload of Figure 7

   The initial state of the Resource Directory is reflected in the
   following request:

   Req: GET /rd-lookup/res?ep=endpoint1

   Res: 2.01 Content
   Payload:
   <coap://local-proxy-old.example.com:5683/sensors/temp>;ct=41;
    rt="temperature"; anchor="coap://spurious.example.com:5683",
   <coap://local-proxy-old.example.com:5683/sensors/light>;ct=41;
     rt="light-lux"; if="sensor";
     anchor="coap://local-proxy-old.example.com:5683"

   The following example shows the registering endpoint changing the
   Base URI to "coaps://new.example.com:5684":

   Req: POST /rd/4521?base=coaps://new.example.com:5684

   Res: 2.04 Changed

   The consecutive query returns:

Req: GET /rd-lookup/res?ep=endpoint1

Res: 2.01 Content
Payload:
<coaps://new.example.com:5684/sensors/temp>;ct=41;rt="temperature";
    anchor="coap://spurious.example.com:5683",
<coaps://new.example.com:5684/sensors/light>;ct=41;rt="light-lux";
    if="sensor"; anchor="coaps://new.example.com:5684",

A.2. Registration Removal

Although RD entries have soft state and will eventually timeout after
their lifetime, the registering endpoint SHOULD explicitly remove an
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.

Removed registrations are implicitly removed from the groups to which
they belong.

The removal request interface is specified as follows:

Interaction: EP -> RD

Method: DELETE

URI Template: {+location}

URI Template Variables:

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

The following response 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 with example location value /rd/4521.

Req: DELETE /rd/4521

Res: 2.02 Deleted

[A.3](). Read Endpoint Links

Some registering endpoints may wish to manage their links as a
collection, and may need to read the current set of links stored in
the registration resource, 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]()

If no links are selected, the Resource Directory SHOULD return an
empty payload.

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 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 response 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, with example location value /rd/4521 and example
   registration payload of Figure 7.

   Req: GET /rd/4521

   Res: 2.01 Content
   Payload:
   </sensors/temp>;ct=41;rt="temperature-c";if="sensor";
   anchor="coap://spurious.example.com:5683",
   </sensors/light>;ct=41;rt="light-lux";if="sensor"

[A.4](#). Update Endpoint Links

   An iPATCH (or PATCH) update ([[RFC8132](#)]) can add, remove or change the
   links of a registration.

   Those operations are out of scope of this document, and will require
   media types suitable for modifying sets of links.

[A.5](#). Endpoint and group lookup

   Endpoint and group lookups result in links to registration resources
   and group resources, respectively. Endpoint registration resources
   are annotated with their endpoint names (ep), sectors (d, if present)
   and registration base URI (base) as well as a constant resource type
   (rt="core.rd-ep"); the lifetime (lt) is not reported. Additional
   endpoint attributes are added as link attributes to their endpoint
   link unless their specification says otherwise.

   Group resources are annotated with their group names (gp), sector (d,
   if present) and multicast address (base, if present) as well as a
   constant resource type (rt="core.rd-gp").

   Serializations derived from Link Format, SHOULD present links to
   groups and endpoints in path-absolute form or, if required, as
   absolute references. (This approach avoids the [RFC6690](#) ambiguities.)

While Endpoint Lookup does expose the registration resources, the RD does not need to make them accessible to clients. Clients SHOULD NOT attempt to dereference or manipulate them.

A Resource Directory can report endpoints or groups in lookup that are not hosted at the same address. Lookup clients MUST be prepared to see arbitrary URIs as registration or group resources in the results and treat them as opaque identifiers; the precise semantics of such links are left to future specifications.

For groups, a Resource Directory as specified here does not provide a lookup mechanism for the resources that can be accessed on a group's multicast address (i.e. no lookup will return links like "<coap://[ff35:30:2001:db8::1]/light>;..." for a group registered with "base=coap://[ff35...]"). Such an additional lookup interface could be specified in an extension document.

The following example shows a client performing an endpoint type (et) lookup with the value oic.d.sensor (which is currently a registered rt value):

Req: GET /rd-lookup/ep?et=oic.d.sensor

Res: 2.05 Content
</rd/1234>;base="coap://[2001:db8:3::127]:61616";ep="node5";
et="oic.d.sensor";ct="40",
</rd/4521>;base="coap://[2001:db8:3::129]:61616";ep="node7";
et="oic.d.sensor";ct="40";d="floor-3"

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

Req: GET /rd-lookup/gp

Res: 2.05 Content
</rd-group/1>;gp="lights1";d="example.com";
        base="coap://[ff35:30:2001:db8::1]",
</rd-group/2>;gp="lights2";d="example.com";
        base="coap://[ff35:30:2001:db8::2]"

The following example shows a client performing a lookup for all groups the endpoint "node1" belongs to:

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

Res: 2.05 Content
</rd-group/1>;gp="lights1"

Appendix B. Web links and the Resource Directory

   Understanding the semantics of a link-format document and its URI
   references is a journey through different documents ([RFC3986]
   defining URIs, [RFC6690] defining link-format documents based on
   [RFC8288] which defines link headers, and [RFC7252] providing the
   transport). This appendix summarizes the mechanisms and semantics at
   play from an entry in ".well-known/core" to a resource lookup.

   This text is primarily aimed at people entering the field of
   Constrained Restful Environments from applications that previously
   did not use web mechanisms.

B.1. A simple example

   Let's start this example with a very simple host, "2001:db8:f0::1".
   A client that follows classical CoAP Discovery ([RFC7252] Section 7),
   sends the following multicast request to learn about neighbours
   supporting resources with resource-type "temperature".

   The client sends a link-local multicast:

   GET coap://[ff02::fd]:5683/.well-known/core?rt=temperature

   RES 2.05 Content
   </temp>;rt=temperature;ct=0

   where the response is sent by the server, "[2001:db8:f0::1]:5683".

   While the client - on the practical or implementation side - can just
   go ahead and create a new request to "[2001:db8:f0::1]:5683" with
   Uri-Path: "temp", the full resolution steps for insertion into and
   retrieval from the RD without any shortcuts are:

B.1.1. Resolving the URIs

   The client parses the single returned record. The link's target
   (sometimes called "href") is ""/temp"", which is a relative URI that
   needs resolving. As long as all involved links follow the
   restrictions set forth for this document (see Appendix B.4), the base
   URI <coap://[ff02::fd]:5683/.well-known/core> is used to resolve the
   reference /temp against.

   The Base URI of the requested resource can be composed from the
   header options of the CoAP GET request by following the steps of
   [RFC7252] section 6.5 (with an addition at the end of 8.2) into
   ""coap://[2001:db8:f0::1]/.well-known/core"".

The record's target is resolved by replacing the path ""/.well-known/
core"" from the Base URI ([section 5.2 [RFC3986]](#)) with the relative
target URI ""/temp"" into ""coap://[2001:db8:f0::1]/temp"".

[B.1.2](#). Interpreting attributes and relations

Some more information but the record's target can be obtained from
the payload: the resource type of the target is "temperature", and
its content type is text/plain (ct=0).

A relation in a web link is a three-part statement that specifies a
named relation between the so-called "context resource" and the
target resource, like "_This page_ has _its table of contents_ at _/
toc.html_". In [[RFC6690](#)] link-format documents, there is an implicit
"host relation" specified with default parameter: rel="hosts".

In our example, the context resource of the link is the URI specified
in the GET request "coap:://[2001:db8:f0::1]/.well-known/core". A
full English expression of the "host relation" is:

'"coap://[2001:db8:f0::1]/.well-known/core" is hosting the resource
"coap://[2001:db8:f0::1]/temp", which is of the resource type
"temperature" and can be accessed using the text/plain content
format.'

[B.2](#). A slightly more complex example

Omitting the "rt=temperature" filter, the discovery query would have
given some more records in the payload:

GET coap://[ff02::fd]:5683/.well-known/core

RES 2.05 Content
</temp>;rt=temperature;ct=0,
</light>;rt=light-lux;ct=0,
</t>;anchor="/sensors/temp";rel=alternate,
<http://www.example.com/sensors/t123>;anchor="/sensors/temp";
    rel="describedby"

Parsing the third record, the client encounters the "anchor"
parameter. It is a URI relative to the Base URI of the request and
is thus resolved to ""coap://[2001:db8:f0::1]/sensors/temp"". That
is the context resource of the link, so the "rel" statement is not
about the target and the Base URI any more, but about the target and
the resolved URI. Thus, the third record could be read as
""coap://[2001:db8:f0::1]/sensors/temp" has an alternate
representation at "coap://[2001:db8:f0::1]/t"".

Following the same resolution steps, the fourth record can be read as
""coap://[2001:db8:f0::1]/sensors/temp" is described by
"http://www.example.com/sensors/t123"".

B.3. Enter the Resource Directory

The resource directory tries to carry the semantics obtainable by
classical CoAP discovery over to the resource lookup interface as
faithfully as possible.

For the following queries, we will assume that the simple host has
used Simple Registration to register at the resource directory that
was announced to it, sending this request from its UDP port
"[2001:db8:f0::1]:6553":

POST coap://[2001:db8:f01::ff]/.well-known/core?ep=simple-host1

The resource directory would have accepted the registration, and
queried the simple host's ".well-known/core" by itself. As a result,
the host is registered as an endpoint in the RD with the name
"simple-host1". The registration is active for 90000 seconds, and
the endpoint registration Base URI is ""coap://[2001:db8:f0::1]/""
following the resolution steps described in Appendix B.1.1. It
should be remarked that the Base URI constructed that way always
yields a URI of the form: scheme://authority without path suffix.

If the client now queries the RD as it would previously have issued a
multicast request, it would go through the RD discovery steps by
fetching "coap://[2001:db8:f0::ff]/.well-known/core?rt=core.rd-
lookup-res", obtain "coap://[2001:db8:f0::ff]/rd-lookup/res" as the
resource lookup endpoint, and issue a request to
"coap://[2001:db8:f0::ff]/rd-lookup/res?rt=temperature" to receive
the following data:

<coap://[2001:db8:f0::1]/temp>;rt=temperature;ct=0;
    anchor="coap://[2001:db8:f0::1]"

This is not _literally_ the same response that it would have received
from a multicast request, but it contains the equivalent statement:

'"coap://[2001:db8:f0::1]" is hosting the resource
"coap://[2001:db8:f0::1]/temp", which is of the resource type
"temperature" and can be accessed using the text/plain content
format.'

(The difference is whether "/" or "/.well-known/core" hosts the
resources, which is subject of ongoing discussion about RFC6690).

Actually, /.well-known/core does NOT host the resource but stores a
URI reference to the resource.

To complete the examples, the client could also query all resources
hosted at the endpoint with the known endpoint name "simple-host1".
A request to "coap://[2001:db8:f0::ff]/rd-lookup/res?ep=simple-host1"
would return

```
<coap://[2001:db8:f0::1]/temp>;rt=temperature;ct=0;
    anchor="coap://[2001:db8:f0::1]",
<coap://[2001:db8:f0::1]/light>;rt=light-lux;ct=0;
    anchor="coap://[2001:db8:f0::1]",
<coap://[2001:db8:f0::1]/t>;
    anchor="coap://[2001:db8:f0::1]/sensors/temp";rel=alternate,
<http://www.example.com/sensors/t123>;
    anchor="coap://[2001:db8:f0::1]/sensors/temp";rel="describedby"
```

All the target and anchor references are already in absolute form
there, which don't need to be resolved any further.

Had the simple host registered with a base= parameter (e.g.
"?ep=simple-host1&base=coap+tcp://simple-host1.example.com"), that
context would have been used to resolve the relative anchor values
instead, giving

```
<coap+tcp://simple-host1.example.com/temp>;rt=temperature;ct=0;
    anchor="coap+tcp://simple-host1.example.com"
```

and analogous records.

B.4. A note on differences between link-format and Link headers

While link-format and Link headers look very similar and are based on
the same model of typed links, there are some differences between
[RFC6690] and [RFC5988], which are dealt with differently:

o "Resolving the target against the anchor": [RFC6690] Section 2.1
   states that the anchor of a link is used as the Base URI against
   which the term inside the angle brackets (the target) is resolved,
   falling back to the resource's URI with paths stripped off (its
   "Origin"). In contrast to that, [RFC8288] Section B.2 describes
   that the anchor is immaterial to the resolution of the target
   reference.

   RFC6690, in the same section, also states that absent anchors set
   the context of the link to the target's URI with its path stripped
   off, while according to [RFC8288] Section 3.2, the context is the
   resource's base URI.

In the context of a Resource Directory, the authors decided to not
let this become an issue by recommending that links in the
Resource Directory be _deserializable_ by either rule set to give
the same results. Note that all examples of [RFC6690], [RFC8288]
and this document comply with that rule.

The Modernized Link Format is introduced in Appendix D to
formalize what it means to apply the ruleset of RFC8288 to Link
Format documents.

o There is no percent encoding in link-format documents.

A link-format document is a UTF-8 encoded string of Unicode
characters and does not have percent encoding, while Link headers
are practically ASCII strings that use percent encoding for non-
ASCII characters, stating the encoding explicitly when required.

For example, while a Link header in a page about a Swedish city
might read

"Link: </temperature/Malm%C3%B6>;rel="live-environment-data""

a link-format document from the same source might describe the
link as

"</temperature/Malmoe>;rel="live-environment-data""

Parsers and producers of link-format and header data need to be
aware of this difference.

Appendix C. Syntax examples for Protocol Negotiation

[This appendix should not show up in a published version of this
document.]

The protocol negotiation that is being worked on in
[I-D.silverajan-core-coap-protocol-negotiation] makes use of the
Resource Directory.

Until that document is update to use the latest resource-directory
specification, here are some examples of protocol negotiation with
the current Resource Directory:

An endpoint could register as follows from its address
[2001:db8:f1::2]:5683:

```
Req: POST coap://rd.example.com/rd?ep=node1
    &at=coap+tcp://[2001:db8:f1::2]
Content-Format: 40
Payload:
</temperature>;ct=0;rt="temperature";if="core.s"

Res: 2.01 Created
Location-Path: /rd/1234
```

An endpoint lookup would just reflect the registered attributes:

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

Res: 2.05 Content
</rd/1234>;ep="node1";base="coap://[2001:db8:f1::2]:5683";
    at="coap+tcp://[2001:db8:f1::2]"
```

A UDP client would then see the following in a resource lookup:

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

Res: 2.05 Content
<coap://[2001:db8:f1::2]/temperature>;ct=0;rt="temperature";
    if="core.s"; anchor="coap://[2001:db8:f1::2]"
```

while a TCP capable client could say:

```
Req: GET /rd-lookup/res?rt=temperature&tt=tcp

Res: 2.05 Content
<coap+tcp://[2001:db8:f1::2]/temperature>;ct=0;rt="temperature";
    if="core.s";anchor="coap+tcp://[2001:db8:f1::2]"
```

Appendix D. Modernized Link Format parsing

The CoRE Link Format as described in [RFC6690] is unsuitable for some
use cases of the Resource Directory, and their resolution scheme is
often misunderstood by developers familiar with [RFC8288].

For the correct application of base URIs, we describe the
interpretation of a Link Format document as a Modernized Link Format.
In Modernized Link Format, the document is processed as in Link
Format, with the exception of Section 2.1 of [RFC6690]:

o The URI-reference inside angle brackets ("<>") describes the
   target URI of the link.

o The context of the link is expressed by the "anchor" parameter.
   If the anchor attribute is absent, it defaults to the empty
   reference ("").

o Both these references are resolved according to Section 5 of
   [RFC3986].

Content formats derived from [RFC6690] which inherit its resolution
rules, like JSON and CBOR link format of [I-D.ietf-core-links-json],
can be interpreted in analogy to that.

For where the Resource Directory is concerned, all common forms of
links (e.g. all the examples of RFC6690) yield identical results.
When interpreting data read from ".well-known/core", differences in
interpretation only affect links where the absent anchor attribute
means "coap://host/" according to RFC6690 and "coap://host/.well-
known/core" according to Modernized Link format; those typically only
occur in conjunction with the vaguely defined implicit "hosts"
relationship.

D.1. For endpoint developers

When developing endpoints, i.e. when generating documents that will
be submitted to a Resource Directory, the differences between
Modernized Link Format and RFC6690 can be ignored as long as all
relative references start with a slash, and any of the following
applies:

o There is no anchor attribute, and the context of the link does not
   matter to the application.

   Example: "</sensors>;ct=40"

o The anchor is a relative reference.

   Example: "</t>;anchor="/sensors/temp";rel="alternate"

o The target is an absolute reference.

   Example: "<http://www.example.com/sensors/t123>;anchor="/sensors/
   temp";rel="describedby""

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

Christian Amsuess (editor)
Hollandstr. 12/4
1020
Austria

Phone: +43-664-9790639
Email: christian@amsuess.com