CoRE   C. Amsüss, Ed.
Internet-Draft
Intended status: Standards Track   Z. Shelby
Expires: 8 September 2021   ARM
                                                            M. Koster
                                                          SmartThings
                                                           C. Bormann
                                            Universitaet Bremen TZI
                                                     P. van der Stok
                                                           consultant
                                                        7 March 2021

                        CoRE Resource Directory
                 draft-ietf-core-resource-directory-28

Abstract

   In many IoT applications, direct discovery of resources is not
   practical due to sleeping nodes, or networks where multicast traffic
   is inefficient. These problems can be solved by employing an entity
   called a Resource Directory (RD), which contains information about
   resources held on other servers, allowing lookups to be performed for
   those resources. The input to an RD is composed of links and the
   output is composed of links constructed from the information stored
   in the RD. This document specifies the web interfaces that an RD
   supports for web servers to discover the RD and to register,
   maintain, lookup and remove information on resources. Furthermore,
   new target attributes useful in conjunction with an RD are defined.

Note to Readers

   Discussion of this document takes place on the CORE Working Group
   mailing list (core@ietf.org), which is archived at
   https://mailarchive.ietf.org/arch/browse/core/
   (https://mailarchive.ietf.org/arch/browse/core/).

   Source for this draft and an issue tracker can be found at
   https://github.com/core-wg/resource-directory (https://github.com/
   core-wg/resource-directory).

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2021.

Copyright Notice

Table of Contents

1. Introduction

   In the work on Constrained RESTful Environments (CoRE), a REST
   architecture suitable for constrained nodes (e.g. with limited RAM
   and ROM [RFC7228]) and networks (e.g. 6LoWPAN [RFC4944]) has been
   established and is used in Internet-of-Things (IoT) or 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 [RFC8288]. 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
   constrained scenarios, direct discovery of resources is not practical
   due to sleeping nodes, or networks where multicast traffic is
   inefficient. These problems can be solved by employing an entity
   called a Resource Directory (RD), which contains information about
   resources held on other servers, allowing lookups to be performed for
   those resources.

   This document specifies the web interfaces that an RD supports for
   web servers to discover the RD and to register, maintain, lookup and
   remove information on resources. Furthermore, new target attributes
   useful in conjunction with an RD 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 BCP
   14 [RFC2119] [RFC8174] when, and only when, they appear in all
   capitals, as shown here.

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], [RFC8288] 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 (RD)
    A web entity that stores information about web resources and implements the REST interfaces defined in this specification for discovery, for the creation, maintenance and removal of registrations, and for lookup of the registered resources.

Sector
    In the context of an RD, a sector is a logical grouping of endpoints.

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

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 RD. 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 RD 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 RD containing registration resources.

   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
      installation events 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. A new IPv6 Neighbor Discovery
      option defined for announcing an RD's address.

3. Architecture and Use Cases

3.1. Principles

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

It provides information about resources hosted by other devices that
could otherwise only be obtained by directly querying the /.well-
known/core resource on these other devices, either by a unicast
request or a multicast request.

Information SHOULD only be stored in the RD if it can be obtained by
querying the described device's /.well-known/core resource directly.

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

3.2. Architecture

The RD architecture is illustrated in Figure 1. An RD is used as a
repository of registrations 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 RD registrations,
and for endpoints to lookup resources from the RD. An RD can be
logically segmented by the use of Sectors.

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

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

An endpoint uses specific interfaces to register, update and remove a
registration. It is also possible for an RD to fetch Web Links from
endpoints and add their contents to its registrations.

At the first registration of an endpoint, 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 registrations.

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

```
                 Registration          Lookup
                   Interface          Interface
    +----+             |                  |
    | EP |----         |                  |
    +----+    ----     |                  |
                  --|-     +------+        |
    +----+          | ----|      |        |    +--------+
    | EP | ---------|-----|  RD  |----|-----| Client |
    +----+          | ----|      |        |    +--------+
                  --|-     +------+        |
    +----+    ----     |                  |
    | CT |----         |                  |
    +----+             |                  |
```
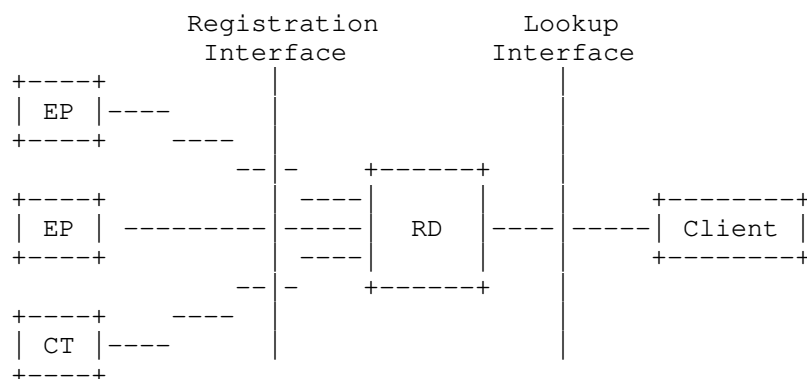
                   Figure 1: The RD architecture.

   A Registrant-EP MAY keep concurrent registrations to more than one RD
   at the same time if explicitly configured to do so, but that is not
   expected to be supported by typical EP implementations. Any such
   registrations are independent of each other. The usual expectation
   when multiple discovery mechanisms or addresses are configured is
   that they constitute a fall-back path for a single registration.

3.3. RD Content Model

   The Entity-Relationship (ER) models shown in Figure 2 and Figure 3
   model the contents of /.well-known/core and the RD 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 an RD. 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    ooooooooo
                        +-----o target o
                             |    ooooooooo
                             |
        oooooooooooo   0+    |
        o    target  o-------+
        o  attribute o       | 0+   oooooo
         oooooooooooo        +-----o rel  o
                             |     oooooo
                             |
                             | 1    ooooooooo
                        +-----o context o
                                  ooooooooo
```
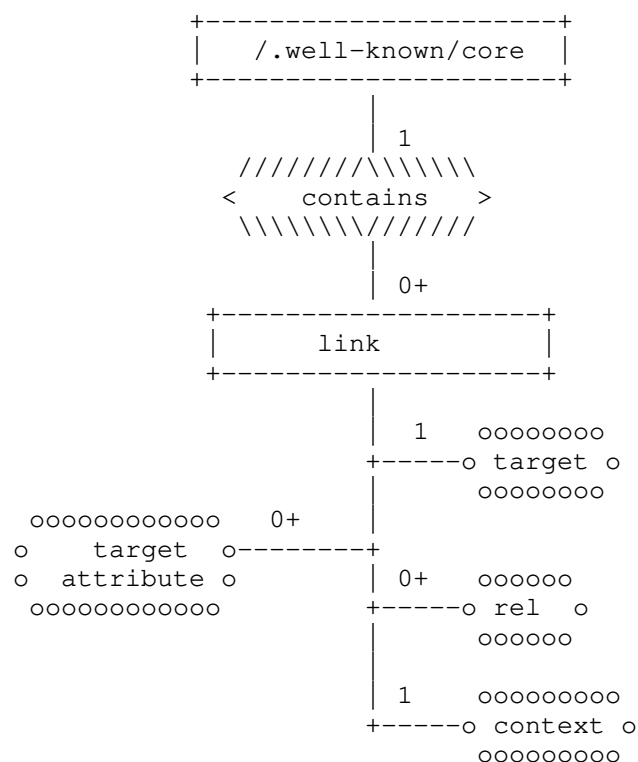
Figure 2: ER Model of the content of /.well-known/core

The model shown in Figure 2 models the contents of /.well-known/core
which contains:

* 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 [RFC8288]):

* 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".

   * 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 and defaults to the Origin of the target (practically:
      the target with its path and later components removed)

   * 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".

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

```
                      +--------------+
                      +     RD       +
                      +--------------+
                             │ 1
                             │
                             │
                             │
                        //////\\\\
                       < contains >
                        \\\\\/////
                             │
                         0+  │
  ooooooo      1  +--------------+
 o  base o-------│ registration │
  ooooooo        +--------------+
                     │         │ 1
                     │         +--------------+
   oooooooo   1  │       /////\\\\         │
  o  href  o----+      < contains >
   oooooooo       │       \\\\\/////
                  │             │
   oooooooo   1  │         0+  │
  o   ep   o----+      +-----------------+
   oooooooo       │     │      link      │
                  │     +-----------------+
   oooooooo  0-1 │             │
  o    d   o----+             │  1   oooooooo
   oooooooo       │         +-----o target o
                  │         │      oooooooo
   oooooooo   1  │         │
  o   lt   o----+   oooooooooo    0+ │
   oooooooo       │  o  target   o-----+
                  │  o attribute o  │ 0+   oooooo
  ooooooooooo 0+ │   oooooooooo    +-----o rel  o
 o  endpoint o----+             │      oooooo
 o attribute o                  │
  ooooooooooo                   │ 1   oooooooo
                                +----o context o
                                     oooooooooo
```
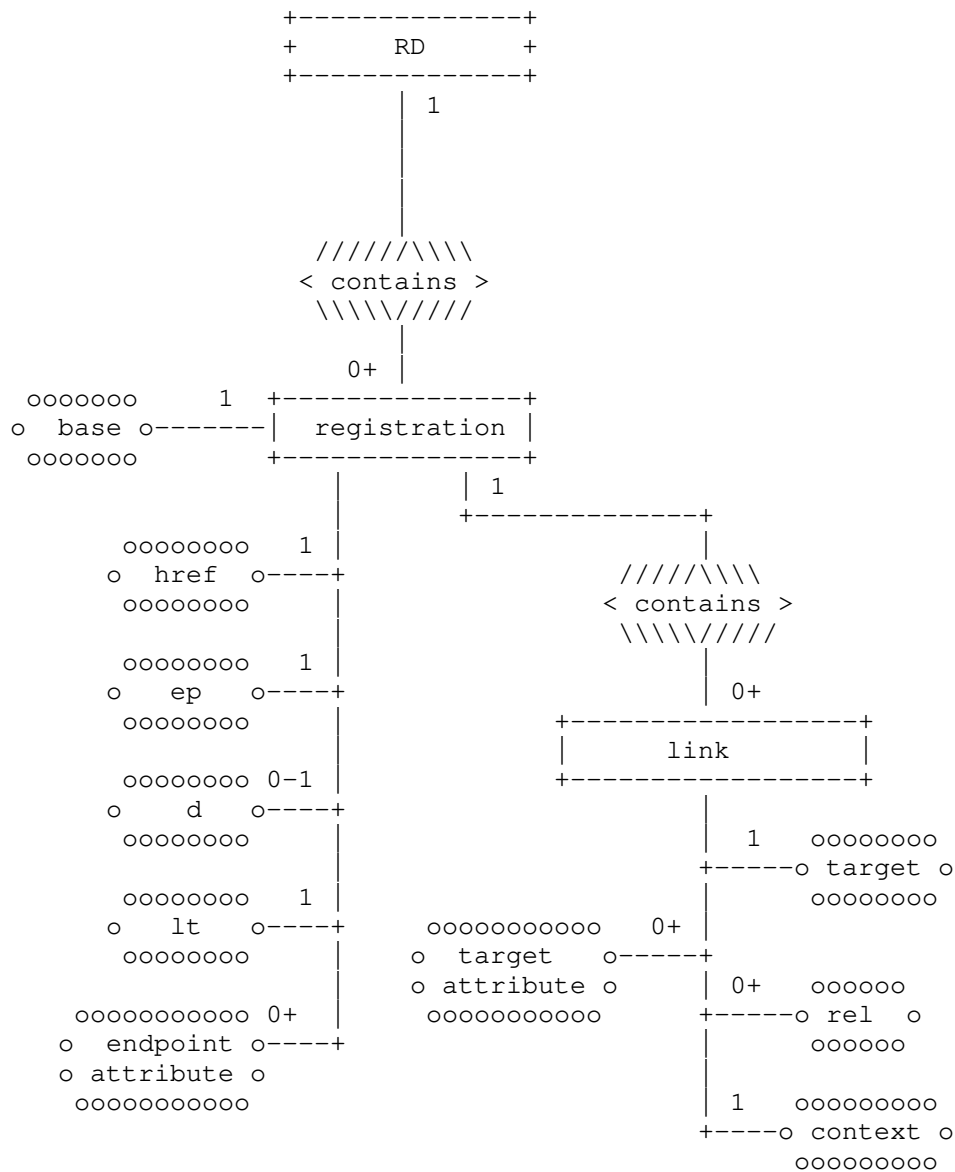
                Figure 3: ER Model of the content of the RD

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

   *   0 to n Registrations of endpoints,

A registration is associated with one endpoint. A registration
defines a set of links as defined for /.well-known/core. A
Registration has six types of attributes:

* an endpoint name ("ep", a Unicode string) unique within a sector

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

* a lifetime ("lt"),

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

* optionally a sector ("d", a Unicode string)

* optional additional endpoint attributes (from Section 9.3)

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 2.

3.4. Link-local addresses and zone identifiers

Registration Base URIs can contain link-local IP addresses. To be
usable across hosts, those cannot be serialized to contain zone
identifiers (see [RFC6874] Section 1).

Link-local addresses can only be used on a single link (therefore RD
servers cannot announce them when queried on a different link), and
lookup clients using them need to keep track of which interface they
got them from.

Therefore, it is advisable in many scenarios to use addresses with
larger scope if available.

3.5. 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. The ambition in such
systems is to build them from reusable components. These speed up

development and deployment, and enable shared use of machines across
different applications. One crucial component of such systems is the
discovery of resources (and thus the endpoints they are hosted on)
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 an RD, 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.

3.6. Use Case: Home and Building Automation

Home and commercial building automation systems can benefit from the
use of IoT 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. Both can use the common RD infrastructure to
establish device interactions efficiently, but can pick security
policies suitable for their needs.

Two phases can be discerned for a network servicing the system: (1)
installation and (2) operation. During the operational phase, the
network is connected to the Internet with a Border Router (e.g. a
6LoWPAN Border Router (6LBR), see [RFC6775]) and the nodes connected
to the network can use the Internet services that are provided by the
Internet Provider or the network administrator. During the
installation phase, the network is completely stand-alone, no Border
Router is connected, and the network only supports the IP
communication between the connected nodes. The installation phase is

usually followed by the operational phase. As an RD's operations
work without hard dependencies on names or addresses, it can be used
for discovery across both phases.

3.7. Use Case: Link Catalogues

   Resources may be shared through data brokers that have no knowledge
   beforehand of who is going to consume the data. An RD 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 an RD. Applications wishing
   to consume the data can use RD Lookup to discover and resolve links
   to the desired resources and endpoints. The RD service need not be
   coupled with the data intermediary service. Mapping of RDs 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 RDs. External
   catalogues that are represented in other formats may be converted to
   common web linking formats for storage and access by RDs. Since it
   is common practice for these to be encoded in URNs [RFC8141], simple
   and lossless structural transforms should generally be sufficient to
   store external metadata in RDs.

   The additional features of an RD 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. Application groups with multicast addresses may be
   defined to support efficient data transport.

4. RD discovery and other interface-independent components

   This and the following sections define the required set of REST
   interfaces between an RD, endpoints and lookup clients. Although the
   examples throughout these sections assume the use of CoAP [RFC7252],
   these REST interfaces can also be realized using HTTP [RFC7230]. The
   multicast discovery and simple registration operations are exceptions
   to that, as they rely on mechanisms unavailable in HTTP. In all
   definitions in these sections, 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.

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

For several operations, interface templates are given in list form;
those describe the operation participants, request codes, URIs,
content formats and outcomes. Sections of those templates contain
normative content about Interaction, Method, URI Template and URI
Template Variables 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.

REST clients (registrant-EPs and CTs during registration and
maintenance, lookup clients, RD servers during simple registrations)
must be prepared to receive any unsuccessful code and act upon it
according to its definition, options and/or payload to the best of
their capabilities, falling back to failing the operation if recovery
is not possible. In particular, they SHOULD retry the request upon
5.03 (Service Unavailable; 503 in HTTP) according to the Max-Age
(Retry-After in HTTP) option, and SHOULD fall back to link-format
when receiving 4.15 (Unsupported Content-Format; 415 in HTTP).

An RD MAY make the information submitted to it available to further
directories (subject to security policies on link confidentiality),
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.

4.1. Finding a Resource Directory

A (re-)starting device may want to find one or more RDs before it can
discover their URIs. Dependent on the operational conditions, one or
more of the techniques below apply.

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

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, as outlined in Section 4.1.2.

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

1. The IPv6 Neighbor Discovery option RDAO Section 4.1.1 can do
    that.

2. When DHCP is in use, 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 RD.

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 an
    RD (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.

When answering a multicast request directed at a link-local group,
the RD may want to respond from a routable address; this makes it
easier for registrants to use one of their own routable addresses for

registration. When [RFC6724] is used for source address selection,
this can be achieved by applying the changes of its Section 10.4,
picking public addresses in its Section 5 Rule 7, and superseding
rule 8 with preferring the source address's precedence.

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.

The following RD discovery mechanisms are recommended:

* In managed networks with border routers that need stand-alone
   operation, the RDAO option is recommended (e.g. operational phase
   described in Section 3.6).

* In managed networks without border router (no Internet services
   available), the use of a preconfigured anycast address is
   recommended (e.g. installation phase described in Section 3.6).

* In networks managed using DNS-SD, the use of DNS-SD for discovery
   as described in Section 4.1.2 is recommended.

The use of multicast discovery in mesh networks is NOT RECOMMENDED.

4.1.1. Resource Directory Address Option (RDAO)

The Resource Directory Address Option (RDAO) carries information
about the address of the RD in RAs (Router Advertisements) of IPv6
Neighbor Discovery (ND), similar to how RDNSS options [RFC8106] are
sent. This information is needed when endpoints cannot discover the
RD with a link-local or realm-local scope multicast address, for
instance 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 RD 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   |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Valid Lifetime                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                        RD Address                             +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
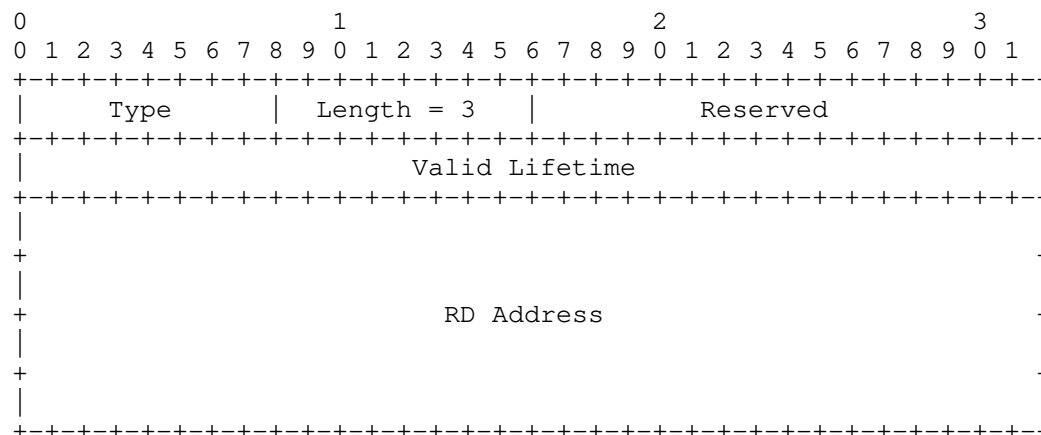
Fields:

Type: TBD38

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

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

Valid Lifetime: 32-bit unsigned integer. The length of
                     time in seconds (relative to
                     the time the packet is received) that
                     this RD address is valid.
                     A value of all zero bits (0x0) indicates
                     that this RD address
                     is not valid anymore.

RD Address: IPv6 address of the RD.

                 Figure 4: Resource Directory Address Option

4.1.2. Using DNS-SD to discover a Resource Directory

   An RD can advertise its presence in DNS-SD [RFC6763] using the
   service name "_core-rd._udp" (for CoAP), "_core-rd-dtls._udp" (for
   CoAP over DTLS), "_core-rd._tcp" (for CoAP over TCP) or "_core-rd-
   tls._tcp" (for CoAP over TLS) defined in this document. (For the
   WebSocket transports of CoAP, no service is defined as DNS-SD is
   typically unavailable in environments where CoAP over WebSockets is
   used).

   The selection of the service indicates the protocol used, and the SRV
   record points the client to a host name and port to use as a starting
   point for the URI discovery steps of Section 4.3.

   This section is a simplified concrete application of the more generic
   mechanism specified in [I-D.ietf-core-rd-dns-sd].

4.2. Payload Content Formats

   RDs implementing this specification MUST support the application/
   link-format content format (ct=40).

   RDs implementing this specification MAY support additional content
   formats.

   Any additional content format supported by an RD implementing this
   specification SHOULD be able to express all the information
   expressible in link-format. It MAY be able to express information
   that is inexpressible in link-format, but those expressions SHOULD be
   avoided where possible.

4.3. 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] after having
   discovered a host as described in Section 4.1.

   Discovery of the RD registration URI 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 URIs for RD 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 9.5).

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

HTTP does not support multicast and consequently only unicast discovery can be supported at the using the HTTP "/.well-known/core" resource.

RDs implementing this 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.

With security policies where the client requires the RD to be authorized to act as an RD, that authorization may be limited to resources on which the authorized RD advertises the adequate resource types. Clients that have obtained links they can not rely on yet can repeat the URI discovery step at the /.well-known/core resource of the indicated host to obtain the resource type information from an authorized source.

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, CT or 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", or "core.rd*"

Accept: absent, application/link-format or any other media type
   representing web links

The following response is expected on this interface:

Success: 2.05 "Content" or 200 "OK" with an application/link-format
   or other web link payload containing one or more matching entries
   for the RD resource.

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
Payload:
</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

             Figure 5: 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 a
CBOR and JSON representation from [I-D.ietf-core-links-json] (which
have no numeric values assigned yet, so they are shown as TBD64 and
TBD504 as in that draft). The RD resource locations /rd, and /rd-
lookup are example values. The server in this example also indicates
that it is capable of providing observation on resource lookups.

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

Res: 2.05 Content
Payload:
</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"

      Figure 6: Example discovery exchange indicating additional
                        content-formats

For maintenance, management and debugging, it can be useful to
identify the components that constitute the RD server. The
identification can be used to find client-server incompatibilities,
supported features, required updates and other aspects. The Well-
Known interface described in Section 4 of [RFC6690] can be used to
find such data.

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
Payload:
<http://software.example.com/shiny-resource-directory/1.0beta1>;
    rel=impl-info

        Figure 7: Example exchange of obtaining implementation
     information, using the relation type currently proposed in the
                      work-in-progress document

Note that depending on the particular server's architecture, such a
link could be anchored at the RD server's root (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. 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] or other representations of
web links, 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 9.3). 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 a registration request targeting a
given (ep, d) value pair:

*   When the (ep, d) value pair of the registration-request is
    different from any existing registration, a new registration is
    generated.

*   When the (ep, d) value pair of the registration-request is equal
    to an existing registration, the content and parameters of the
    existing registration are replaced with the content of the
    registration request. Like the later changes to registration
    resources, security policies (Section 7) usually require such
    requests to come from the same device.

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.

For media types to which Appendix C applies (i.e. documents in
application/link-format), request bodies MUST be expressed in Limited
Link Format.

The registration request interface is specified as follows:

Interaction: EP or CT -> 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.

     As the endpoint name is a Unicode string, it is encoded in
     UTF-8 (and possibly pct-encoded) during variable expansion (see
     [RFC6570] Section 3.2.1). The endpoint name MUST NOT contain
     any character in the inclusive ranges 0-31 or 127-159.

     The maximum length of this parameter is 63 UTF-8 encoded bytes.

If the RD is configured to recognize the endpoint to be
authorized to use exactly one endpoint name, the RD assigns
that name. In that case, giving the endpoint name becomes
optional for the client; if the client gives any other endpoint
name, it is not authorized to perform the registration.

d := Sector (optional). The sector to
which this endpoint belongs. When this parameter is not
present, the RD MAY associate the endpoint with a configured
default sector (possibly based on the endpoint's authorization)
or leave it empty.

The sector is encoded like the ep parameter, and is limited to
63 UTF-8 encoded bytes as well.

lt := Lifetime (optional). Lifetime of the
registration in seconds. Range of 1-4294967295. 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. The Base URI is consecutively constructed by
concatenating the used protocol'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.

A registrant that cannot be reached by potential lookup clients

at the address it registers from (e.g. because it is behind
some form of Network Address Translation (NAT)) MUST provide a
reachable base address with its registration.

If the Base URI contains a link-local IP literal, it MUST NOT
contain a Zone Identifier, and MUST be local to the link on
which the registration request is received.

Endpoints that register with a base that contains a path
component cannot efficiently express their registrations in
Limited Link Format (Appendix C). Those applications should
use different representations of links to which Appendix C is
not applicable (e.g. [I-D.hartke-t2trg-coral]).

extra-attrs := Additional registration
attributes (optional). The endpoint can pass any parameter
registered at Section 9.3 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 or any other indicated media
   type representing web links

The following response is expected on this interface:

Success: 2.01 "Created" or 201 "Created". The Location-Path option
   or Location header field 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.

If the registration fails, including request timeouts, or if delays
from Service Unavailable responses with Max-Age or Retry-After
accumulate to exceed the registrant's configured timeouts, 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 5.

```
Req: POST coap://rd.example.com/rd?ep=node1
Content-Format: 40
Payload:
</sensors/temp>;rt=temperature-c;if=sensor,
<http://www.example.com/sensors/temp>;
  anchor="/sensors/temp";rel=describedby

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

                 Figure 8: Example registration payload

An RD may optionally support HTTP. Here is an example of almost the
same registration operation above, when done using HTTP.

```
Req:
POST /rd?ep=node1&base=http://[2001:db8:1::1] HTTP/1.1
Host: rd.example.com
Content-Type: application/link-format

</sensors/temp>;rt=temperature-c;if=sensor,
<http://www.example.com/sensors/temp>;
  anchor="/sensors/temp";rel=describedby

Res:
HTTP/1.1 201 Created
Location: /rd/4521
```

     Figure 9: Example registration payload as expressed using HTTP

5.1. Simple Registration

   Not all endpoints hosting resources are expected to know how to
   upload links to an RD as described in Section 5. 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 C).

   * The registrant-ep finds one or more addresses of the directory
      server as described in Section 4.1.

   * The registrant-ep sends (and regularly refreshes with) a POST
      request to the "/.well-known/rd" 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 with a regular registration per
      Section 5. The registration base URI of the registration is taken
      from the registrant-ep's network address (as is default with
      regular registrations).

      Example request from registrant-EP to RD (unanswered until the
      next step):

   Req: POST /.well-known/rd?lt=6000&ep=node1
   (No payload)

      Figure 10: First half example exchange of a simple registration

   * The RD queries the registrant-ep's discovery resource to determine
      the success of the operation. It SHOULD keep a cache of the
      discovery resource and not query it again as long as it is fresh.

      Example request from the RD to the registrant-EP:

```
Req: GET /.well-known/core
Accept: 40

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

   Figure 11: Example exchange of the RD querying the simple endpoint

With this response, the RD would answer the previous step's request:

Res: 2.04 Changed

   Figure 12: Second half example exchange of a simple registration

The sequence of fetching the registration content before sending a
successful response was chosen to make responses reliable, and the
point about caching was chosen to still allow very constrained
registrants. Registrants MUST be able to serve a GET request to
"/.well-known/core" after having requested registration. Constrained
devices MAY regard the initial request as temporarily failed when
they need RAM occupied by their own request to serve the RD's GET,
and retry later when the RD already has a cached representation of
their discovery resources. Then, the RD can reply immediately and
the registrant can receive the response.

The simple registration request interface is specified as follows:

Interaction: EP -> RD

Method: POST

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

URI Template Variables are as they are for registration in Section 5.
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. For some time during this document's
development, the URI template "/.well-known/core{?ep,...}" has been
in use instead.

The following response is expected on this interface:

Success: 2.04 "Changed".

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 is expected on this interface:

Success: 2.05 "Content".

When the RD uses any authorization credentials to access the
endpoint's discovery resource, or when it is deployed in a location
where third parties might reach it but not the endpoint, it SHOULD
verify that the apparent registrant-ep intends to register with the
given registration parameters before revealing the obtained discovery
information to lookup clients. An easy way to do that is to verify
the simple registration request's sender address using the Echo
option as described in [I-D.ietf-core-echo-request-tag] Section 2.4.

The RD MUST delete registrations created by simple registration after
the expiration of their lifetime. Additional operations on the
registration resource cannot be executed because no registration
location is returned.

5.2. 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 RD can be
filled by a third party device, called a Commissioning Tool (CT).
The commissioning tool can fill the RD 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.

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.

5.3. Operations on the Registration Resource

   This section describes how the registering endpoint can maintain the
   registrations that it created. The registering endpoint can be the
   registrant-ep or the CT. The registrations are resources of the RD.

   An endpoint should not use this interface for registrations that it
   did not create. This is usually enforced by security policies, which
   in general require equivalent credentials for creation of and
   operations on a registration.

   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. If the
   registration resource is removed, the corresponding endpoint will
   need to be re-registered.

   The registration resource may also be used cancel the registration
   using DELETE, and to perform further operations beyond the scope of
   this specification.

   Operations on the registration resource are sensitive to reordering;
   Section 5.3.4 describes how order is restored.

   The operations on the registration resource are described below.

5.3.1. 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 registration parameters like lifetime, base URI
   or others. 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 are included as query parameters in an
   update operation as in Section 5.

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

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 Section 5.3.3).

The update registration request interface is specified as follows:

Interaction: EP or CT -> RD

Method: POST

URI Template: {+location}{?lt,base,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 1-4294967295. 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, and is subject to the same restrictions as in the registration.

> 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.

> 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.

Note that this default behavior does not allow removing an
endpoint attribute in an update. For attributes whose
functionality depends on the endpoints' ability to remove them
in an update, it can make sense to define a value whose
presence is equivalent to the absence of a value. As an
alternative, an extension can define different updating rules
for their attributes. That necessitates either discovery of
whether the RD is aware of that extension, or tolerating the
default behavior.

Content-Format: none (no payload)

The following responses are expected on this interface:

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

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

If the registration update fails in any way, including "Not Found"
and request timeouts, or if the time indicated in a Service
Unavailable Max-Age/Retry-After exceeds the remaining lifetime, the
registering endpoint SHOULD attempt registration again.

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

Req: POST /rd/4521

Res: 2.04 Changed

                  Figure 13: Example update of a registration

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:

* endpoint name (ep)=endpoint1

* lifetime (lt)=500

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

   *   payload of Figure 8

   The initial state of the RD is reflected in the following request:

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

   Res: 2.05 Content
   Payload:
   <coap://local-proxy-old.example.com/sensors/temp>;
       rt=temperature-c;if=sensor,
   <http://www.example.com/sensors/temp>;
       anchor="coap://local-proxy-old.example.com/sensors/temp";
       rel=describedby

       Figure 14: Example lookup before a change to the base address

   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

   Res: 2.04 Changed

    Figure 15: Example registration update that changes the base address

   The consecutive query returns:

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

   Res: 2.05 Content
   Payload:
   <coaps://new.example.com/sensors/temp>;
       rt=temperature-c;if=sensor,
   <http://www.example.com/sensors/temp>;
       anchor="coaps://new.example.com/sensors/temp";
       rel=describedby

        Figure 16: Example lookup after a change to the base address

5.3.2. Registration Removal

   Although RD registrations 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.

   The removal request interface is specified as follows:

   Interaction: EP or CT -> 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 responses are expected on this interface:

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

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

   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

                 Figure 17: Example of a registration removal

5.3.3. Further operations

   Additional operations on the registration can be specified in future
   documents, for example:

   * Send iPATCH (or PATCH) updates ([RFC8132]) to add, remove or
      change the links of a registration.

   * Use GET to read the currently stored set of links in a
      registration resource.

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

5.3.4. Request freshness

Some security mechanisms usable with an RD allow out of order request
processing, or do not even mandate replay protection at all. The RD
needs to ensure that operations on the registration resource are
executed in an order that does not distort the client's intentions.

This ordering of operations is expressed in terms of freshness as
defined in [I-D.ietf-core-echo-request-tag]. Requests that alter a
resource's state need to be fresh relative to the latest request that
altered that state in a conflicting way.

An RD SHOULD determine a request's freshness, and MUST use the Echo
option if it requires request freshness and can not determine the it
in any other way. An endpoint MUST support the use of the Echo
option. (One reason why an RD would not require freshness is when no
relevant registration properties are covered by is security
policies.)

5.3.4.1. Efficient use of Echo by an RD

To keep latency and traffic added by the freshness requirements to a
minimum, RDs should avoid naive (sufficient but inefficient)
freshness criteria.

Some simple mechanisms the RD can employ are:

* State counter. The RD can keep a monotonous counter that
   increments whenever a registration changes. For every
   registration resource, it stores the post-increment value of that
   resource's last change. Requests altering them need to have at
   least that value encoded in their Echo option, and are otherwise
   rejected with a 4.01 Unauthorized and the current counter value as
   the Echo value. If other applications on the same server use Echo
   as well, that encoding may include a prefix indicating that it
   pertains to the RD's counter.

   The value associated with a resource needs to be kept across the
   removal of registrations if the same registration resource is to
   be reused.

   The counter can be reset (and the values of removed resources
   forgotten) when all previous security associations are reset.

This is the "Persistent Counter" method of
[I-D.ietf-core-echo-request-tag] Appendix A.

* Preemptive Echo values. The current state counter can be sent in
   an Echo option not only when requests are rejected with 4.01
   Unauthorized, but also with successful responses. Thus, clients
   can be provided with Echo values sufficient for their next request
   on a regular basis.

   While endpoints may discard received Echo values at leisure
   between requests, they are encouraged to retain these values for
   the next request to avoid additional round trips.

* If the RD can ensure that only one security association has
   modifying access to any registration at any given time, and that
   security association provides order on the requests, that order is
   sufficient to show request freshness.

5.3.4.2. Examples of Echo usage

Figure 18 shows the interactions of an endpoint that has forgotten
the server's latest Echo value and temporarily reduces its
registration lifetime:

Req: POST /rd/4521?lt=7200

Res: 4.01 Unauthorized
Echo: 0x0123

(EP tries again immediately)

Req: POST /rd/4521?lt=7200
Echo: 0x0123

Res: 2.04 Changed
Echo: 0x0124

(Later the EP regains its confidence in its long-term reachability)

Req: POST /rd/4521?lt=90000
Echo: 0x0124

Res: 2.04 Changed
Echo: 0x0247

              Figure 18: Example update of a registration

The other examples do not show Echo options for simplicity, and
because they lack the context for any example values to have meaning.

6. 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. JSON or CBOR link format
[I-D.ietf-core-links-json]) or using more advanced interfaces (e.g.
supporting context or semantic based lookup) on different resources
that are discovered independently.

RD Lookup allows lookups for 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, 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 |
+-------------+--------------------+-----------+
```

Table 1: Lookup Types

6.1. Resource lookup

Resource lookup results in links that are semantically equivalent to
the links submitted to the RD by the registrant. The links and link
parameters returned by the lookup are equal to the originally
submitted ones, except that the target reference is fully resolved,
and that the anchor reference is fully resolved if it is present in
the lookup result at all.

Links that did not have an anchor attribute in the registration are
returned without an anchor attribute. Links of which href or anchor
was submitted as a (full) URI are returned with the respective
attribute unmodified.

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

If the base URI of a registration contains a link-local address, the
RD MUST NOT show its links unless the lookup was made from the link
on which the registered endpoint can be reached. The RD MUST NOT
include zone identifiers in the resolved URIs.

6.2. 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 in alternate content-formats (e.g. from
[I-D.ietf-core-links-json]).

Multiple search criteria MAY be included in a lookup. All included
criteria MUST match for a link to be returned. The RD 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 "relation-types" (in the link-format ABNF) match if the search
value matches any of their values (see Section 4.1 of [RFC6690]; e.g.
"?if=tag:example.net,2020:sensor" matches ";if="example.regname
tag:example.net,2020:sensor";"). A resource link also matches a
search criterion if its endpoint would match the criterion, and vice
versa, an endpoint link matches a search criterion if any of its
resource links matches it.

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. 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 endpoints SHOULD be expressed in path-absolute form if possible
and MUST be expressed in URI form otherwise; the RD SHOULD recognize
either. The "anchor" attribute is usable for resource lookups, and,
if queried, MUST be in URI form as well.

Additional query parameters "page" and "count" 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. Unlike block-wise transfer of a compelte
result set, these parameters ensure that each chunk of results can be
interpreted on its own. This simplifies the processing, but can
result in duplicate or missed items when coinciding with changes from
the registration interface.

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 4.3.

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

      The search criteria are an associative array, expressed in a
      form-style query as per the URI template (see [RFC6570]
      Sections 2.4.2 and 3.2.8)

                        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.

Accept: absent, application/link-format or any other indicated media
   type representing web links

The following responses codes are defined for this interface:

Success: 2.05 "Content" or 200 "OK" with an "application/link-
   format" or other web link payload containing matching entries for
   the lookup.

   The payload can contain zero links (which is an empty payload in
   [RFC6690] link format, but could also be "[]" in JSON based
   formats), indicating that no entities matched the request.

6.3. 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 5:

Req: GET /rd-lookup/res?rt=tag:example.org,2020:temperature

Res: 2.05 Content
Payload:
<coap://[2001:db8:3::123]:61616/temp>;
    rt="tag:example.org,2020:temperature"

                      Figure 19: Example a resource lookup

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

```
Req: GET /rd-lookup/res?rt=tag:example.org,2020: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="tag:example.org,2020:light",
<coap://[2001:db8:3::124]/south>;rt="tag:example.org,2020:light",
<coap://[2001:db8:3::124]/east>;rt="tag:example.org,2020:light"
```

Figure 20: Example an observing resource lookup

The following example shows a client performing a paginated resource
lookup

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

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

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

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

Figure 21: Examples of paginated resource lookup

The following example shows a client performing a lookup of all
resources 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 response of section 5 of
[RFC6690].

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

Req: GET /rd-lookup/res?et=tag:example.com,2020:platform

Res: 2.05 Content
Payload:
<coap://sensor1.example.com/sensors>;ct=40;title="Sensor Index",
<coap://sensor1.example.com/sensors/temp>;rt=temperature-c;if=sensor,
<coap://sensor1.example.com/sensors/light>;rt=light-lux;if=sensor,
<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",
<coap://sensor2.example.com/sensors/temp>;rt=temperature-c;if=sensor,
<coap://sensor2.example.com/sensors/light>;rt=light-lux;if=sensor,
<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"

        Figure 22: Example of resource lookup from multiple endpoints

6.4. Endpoint lookup

The endpoint lookup returns links to and information about
registration resources, which themselves can only be manipulated by
the registering endpoint.

Endpoint registration resources are annotated with their endpoint
names (ep), sectors (d, if present) and registration base URI (base;
reports the registrant-ep's address if no explicit base was given) as
well as a constant resource type (rt="core.rd-ep"); the lifetime (lt)
is not reported. Additional endpoint attributes are added as target
attributes to their endpoint link unless their specification says
otherwise.

Links to endpoints SHOULD be presented in path-absolute form or, if
required, as (full) URIs. (This ensures that the output conforms to
Limited Link Format as described in Appendix C.)

Base addresses that contain link-local addresses MUST NOT include
zone identifiers, and such registrations MUST NOT be shown unless the
lookup was made from the same link from which the registration was
made.

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.

An RD can report registrations in lookup whose URI scheme and
authority differ from the lookup resource's. Lookup clients MUST be
prepared to see arbitrary URIs as registration resources in the
results and treat them as opaque identifiers; the precise semantics
of such links are left to future specifications.

The following example shows a client performing an endpoint lookup
limited to endpoints of endpoint type
"tag:example.com,2020:platform":

Req: GET /rd-lookup/ep?et=tag:example.com,2020:platform

Res: 2.05 Content
Payload:
</rd/1234>;base="coap://[2001:db8:3::127]:61616";ep=node5;
    et="tag:example.com,2020:platform";ct=40;rt=core.rd-ep,
</rd/4521>;base="coap://[2001:db8:3::129]:61616";ep=node7;
    et="tag:example.com,2020:platform";ct=40;d=floor-3;
    rt=core.rd-ep

                Figure 23: Examples of endpoint lookup

7. Security policies

   The security policies that are applicable to an RD strongly depend on
   the application, and are not set out normatively here.

   This section provides a list of aspects that applications should
   consider when describing their use of the RD, without claiming to
   cover all cases. It is using terminology of
   [I-D.ietf-ace-oauth-authz], in which the RD acts as the Resource
   Server (RS), and both registrant-eps and lookup clients act as
   Clients (C) with support from an Authorization Server (AS), without
   the intention of ruling out other (e.g. certificate / public-key
   infrastructure (PKI) based) schemes.

Any, all or none of the below can apply to an application. Which are
relevant depends on its protection objectives.

Security policies are set by configuration of the RD, or by choice of
the implementation. Lookup clients (and, where relevant, endpoints)
can only trust an RD to uphold them if it is authenticated, and
authorized to serve as an RD according to the application's
requirements.

7.1. Endpoint name

Whenever an RD needs to provide trustworthy results to clients doing
endpoint lookup, or resource lookup with filtering on the endpoint
name, the RD must ensure that the registrant is authorized to use the
given endpoint name. This applies both to registration and later to
operations on the registration resource. It is immaterial whether
the client is the registrant-ep itself or a CT is doing the
registration: The RD cannot tell the difference, and CTs may use
authorization credentials authorizing only operations on that
particular endpoint name, or a wider range of endpoint names.

It is up to the concrete security policy to describe how endpoint
name and sector are transported when certificates are used. For
example, it may describe how SubjectAltName dNSName entries are
mapped to endpoint and domain names.

7.1.1. Random endpoint names

Conversely, in applications where the RD does not check the endpoint
name, the authorized registering endpoint can generate a random
number (or string) that identifies the endpoint. The RD should then
remember unique properties of the registrant, associate them with the
registration for as long as its registration resource is active
(which may be longer than the registration's lifetime), and require
the same properties for operations on the registration resource.

Registrants that are prepared to pick a different identifier when
their initial attempt (or attempts, in the unlikely case of two
subsequent collisions) at registration is unauthorized should pick an
identifier at least twice as long as the expected number of
registrants; registrants without such a recovery options should pick
significantly longer endpoint names (e.g. using UUID URNs [RFC4122]).

7.2. Entered resources

When lookup clients expect that certain types of links can only
originate from certain endpoints, then the RD needs to apply
filtering to the links an endpoint may register.

For example, if clients use an RD to find a server that provides
firmware updates, then any registrant that wants to register (or
update) links to firmware sources will need to provide suitable
credentials to do so, independently of its endpoint name.

Note that the impact of having undesirable links in the RD depends on
the application: if the client requires the firmware server to
present credentials as a firmware server, a fraudulent link's impact
is limited to the client revealing its intention to obtain updates
and slowing down the client until it finds a legitimate firmware
server; if the client accepts any credentials from the server as long
as they fit the provided URI, the impact is larger.

An RD may also require that links are only registered if the
registrant is authorized to publish information about the anchor (or
even target) of the link. One way to do this is to demand that the
registrant present the same credentials as a client that they'd need
to present if contacted as a server at the resources' URI, which may
include using the address and port that are part of the URI. Such a
restriction places severe practical limitations on the links that can
be registered.

As above, the impact of undesirable links depends on the extent to
which the lookup client relies on the RD. To avoid the limitations,
RD applications should consider prescribing that lookup clients only
use the discovered information as hints, and describe which pieces of
information need to be verified because they impact the application's
security. A straightforward way to verify such information is to
request it again from an authorized server, typically the one that
hosts the target resource. That similar to what happens in
Section 4.3 when the URI discovery step is repeated.

7.3. Link confidentiality

When registrants publish information in the RD that is not available
to any client that would query the registrant's /.well-known/core
interface, or when lookups to that interface are subject so stricter
firewalling than lookups to the RD, the RD may need to limit which
lookup clients may access the information.

In this case, the endpoint (and not the lookup clients) needs to be
careful to check the RD's authorization. The RD needs to check any
lookup client's authorization before revealing information directly
(in resource lookup) or indirectly (when using it to satisfy a
resource lookup search criterion).

7.4. Segmentation

   Within a single RD, different security policies can apply.

   One example of this are multi-tenant deployments separated by the
   sector (d) parameter. Some sectors might apply limitations on the
   endpoint names available, while others use a random identifier
   approach to endpoint names and place limits on the entered links
   based on their attributes instead.

   Care must be taken in such setups to determine the applicable access
   control measures to each operation. One easy way to do that is to
   mandate the use of the sector parameter on all operations, as no
   credentials are suitable for operations across sector borders anyway.

7.5. First-Come-First-Remembered: A default policy

   The First-Come-First-Remembered policy is provided both as a
   reference example for a security policy definition, and as a policy
   that implementations may choose to use as default policy in absence
   of other configuration. It is designed to enable efficient discovery
   operations even in ad-hoc settings.

   Under this policy, the RD accepts registrations for any endpoint name
   that is not assigned to an active registration resource, and only
   accepts registration updates from the same endpoint. The policy is
   minimal in that towards lookup clients it does not make any of the
   claims of Section 7.2 and Section 7.3, and its claims on Section 7.1
   are limited to the lifetime of that endpoint's registration. It
   does, however, guarantee towards any endpoint that for the duration
   of its registration, its links will be discoverable on the RD.

   When a registration or operation is attempted, the RD MUST determine
   the client's subject name or public key:

   * If the client's credentials indicate any subject name that is
      certified by any authority which the RD recognizes (which may be
      the system's trust anchor store), all such subject names are
      stored. With CWT or JWT based credentials (as common with ACE),
      the Subject (sub) claim is stored as a single name, if it exists.
      With X.509 certificates, the Common Name (CN) and the complete
      list of SubjectAltName entries are stored. In both cases, the
      authority that certified the claim is stored along with the
      subject, as the latter may only be locally unique.

* Otherwise, if the client proves possession of a private key, the
   matching public key is stored. This applies both to raw public
   keys and to the public keys indicated in certificates that failed
   the above authority check.

* If neither is present, a reference to the security session itself
   is stored. With (D)TLS, that is the connection itself, or the
   session resumption information if available. With OSCORE, that is
   the security context.

As part of the registration operation, that information is stored
along with the registration resource.

The RD MUST accept all registrations whose registration resource is
not already active, as long as they are made using a security layer
supported by the RD.

Any operation on a registration resource, including registrations
that lead to an existing registration resource, MUST be rejected by
the RD unless all the stored information is found in the new
request's credentials.

Note that even though subject names are compared in this policy, they
are never directly compared to endpoint names, and an endpoint can
not expect to "own" any particular endpoint name outside of an active
registration -- even if a certificate says so. It is an accepted
shortcoming of this approach that the endpoint has no indication of
whether the RD remembers it by its subject name or public key;
recognition by subject happens on a best-effort base (given the RD
may not recognize any authority). Clients MUST be prepared to pick a
different endpoint name when rejected by the RD initially or after a
change in their credentials; picking an endpoint name as per
Section 7.1.1 is an easy option for that.

For this policy to be usable without configuration, clients should
not set a sector name in their registrations. An RD can set a
default sector name for registrations accepted under this policy,
which is useful especially in a segmented setup where different
policies apply to different sectors. The configuration of such a
behavior, as well as any other configuration applicable to such an RD
(i.e. the set of recognized authorities) is out of scope for this
document.

8. Security Considerations

   The security considerations as described in Section 5 of [RFC8288]
   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].

   Access that is limited or affects sensitive data SHOULD be protected,
   e.g. using (D)TLS or OSCORE ([RFC8613]; which aspects of the RD this
   affects depends on the security policies of the application (see
   Section 7).

8.1. Discovery

   Most steps in discovery of the RD, and possibly its resources, are
   not covered by CoAP's security mechanisms. This will not endanger
   the security properties of the registrations and lookup itself (where
   the client requires authorization of the RD if it expects any
   security properties of the operation), but may leak the client's
   intention to third parties, and allow them to slow down the process.

   To mitigate that, clients can retain the RD's address, use secure
   discovery options like configured addresses, and send queries for RDs
   in a very general form ("?rt=core.rd*" rather than "?rt=core.rd-
   lookup-ep").

8.2. Endpoint Identification and Authentication

   An Endpoint (name, sector) pair is unique within the set of endpoints
   registered 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 an RD 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.

Endpoint authorization needs to be checked on registration and
registration resource operations independently of whether there are
configured requirements on the credentials for a given endpoint name
(and sector; Section 7.1) or whether arbitrary names are accepted
(Section 7.1.1).

Simple registration could be used to circumvent address-based access
control: An attacker would send a simple registration request with
the victim's address as source address, and later look up the
victim's /.well-known/core content in the RD. Mitigation for this is
recommended in Section 5.1.

The registration resource path is visible to any client that is
allowed endpoint lookup, and can be extracted by resource lookup
clients as well. The same goes for registration attributes that are
shown as target attributes or lookup attributes. The RD needs to
consider this in the choice of registration resource paths, and
administrators or endpoint in their choice of attributes.

8.3. Access Control

Access control SHOULD be performed separately for the RD registration
and Lookup 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.

The precise access controls necessary (and the consequences of
failure to enforce them) depend on the protection objectives of the
application and the security policies (Section 7) derived from them.

8.4. Denial of Service Attacks

Services that run over UDP unprotected are vulnerable to unknowingly
amplify and distribute a DoS attack as UDP does not require return
routability check. Since RD lookup responses can be significantly
larger than requests, RDs are prone to this.

[RFC7252] describes this at length in its Section 11.3, including
some mitigation by using small block sizes in responses. The
upcoming [I-D.ietf-core-echo-request-tag] updates that by describing
a source address verification mechanism using the Echo option.

[If this document is published together with or after I-D.ietf-core-
echo-request-tag, the above paragraph is replaced with the following:

[RFC7252] describes this at length in its Section 11.3, and
[I-D.ietf-core-echo-request-tag] (which updates the former)
recommends using the Echo option to verify the request's source
address.

]

8.5. Skipping freshness checks

When RD based applications are built in which request freshness
checks are not performed, these concerns need to be balanced:

* When alterations to registration attributes are reordered, an
   attacker may create any combination of attributes ever set, with
   the attack difficulty determined by the security layer's replay
   properties.

   For example, if Figure 18 were conducted without freshness
   assurances, an attacker could later reset the lifetime back to
   7200. Thus, the device is made unreachable to lookup clients.

* When registration updates without query parameters (which just
   serve to restart the lifetime) can be reordered, an attacker can
   use intercepted messages to give the appearance of the device
   being alive to the RD.

   This is unacceptable when when the RD's security policy promises
   reachability of endpoints (e.g. when disappearing devices would
   trigger further investigation), but may be acceptable with other
   policies.

9. IANA Considerations

9.1. 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 RD| RFCTHIS     |
|                  |                            | Section 4.3 |
+------------------+----------------------------+-------------+
| core.rd-lookup-res | Resource lookup of an RD | RFCTHIS     |
|                  |                            | Section 4.3 |
+------------------+----------------------------+-------------+
| core.rd-lookup-ep | Endpoint lookup of an RD  | RFCTHIS     |
|                  |                            | Section 4.3 |
+------------------+----------------------------+-------------+
| core.rd-ep       | Endpoint resource of an RD | RFCTHIS     |
|                  |                            | Section 6   |
+------------------+----------------------------+-------------+
```

                            Table 2

9.2. IPv6 ND Resource Directory Address Option

   This document registers one new ND option type under the sub-registry
   "IPv6 Neighbor Discovery Option Formats" of the "Internet Control
   Message Protocol version 6 (ICMPv6) Parameters" registry:

   * Resource Directory Address Option (TBD38)

   [The RFC editor is asked to replace TBD38 with the assigned number
   in the document; the value 38 is suggested.]

9.3. RD Parameter Registry

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

   Each entry in the registry must include

   * the human readable name of the parameter,

   * the short name as used in query parameters or target attributes,

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

   * syntax and validity requirements if any,

   * a description,

   * and a link to reference documentation.

   The query parameter MUST be both a valid URI query key [RFC3986] and
   a token as used in [RFC8288].

   The description must give details on whether the parameter can be
   updated, and how it is 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 Name	ep	Unicode*	RLA	Name of the endpoint
Lifetime	lt	1-4294967295	R	Lifetime of the registration in seconds
Sector	d	Unicode*	RLA	Sector to which this endpoint belongs
Registration Base URI	base	URI	RLA	The scheme, address and port and path at which this server is available
Page	page	Integer	L	Used for pagination
Count	count	Integer	L	Used for pagination
Endpoint Type	et	Section 9.3.1	RLA	Semantic type of the endpoint (see Section 9.4)

Table 3: RD Parameters

(Short: Short name used in query parameters or target attributes.
Validity: Unicode* = 63 Bytes of UTF-8 encoded Unicode, with no
control characters as per Section 5. Use: R = used at registration,
L = used at lookup, A = expressed in target 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.
All their reference documentation entries point to 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 target 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] target attribute).

9.3.1. Full description of the "Endpoint Type" RD 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,
RDs implementing this specification automatically support correct
filtering in the lookup interfaces from the rules for unknown
endpoint attributes.

9.4. "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:

* The values MUST be related to the purpose described in
   Section 9.3.1.

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

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

The registry initially contains one value:

   * "core.rd-group": An application group as described in Appendix A.

9.5. Multicast Address Registration

   IANA is asked to assign the following multicast addresses for use by
   CoAP nodes:

   IPv4 -- "all CoRE Resource Directories" address MCD2 (suggestion:
   224.0.1.189), from the "IPv4 Multicast Address Space Registry". 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)
   [RFC5771].

   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.

   [The RFC editor is asked to replace MCD1 and MCD2 with the assigned
   addresses throughout the document.]

9.6. Well-Known URIs

   IANA is asked to permanently register the URI suffix "rd" in the
   "Well-Known URIs" registry. The change controller is the IETF, this
   document is the reference.

9.7. Service Names and Transport Protocol Port Number Registry

   IANA is asked to enter four new items into the Service Names and
   Transport Protocol Port Number Registry:

   * Service name: "core-rd", Protocol: "udp", Description: "Resource
      Directory accessed using CoAP"

   * Service name "core-rd-dtls", Protocol: "udp", Description:
      "Resource Directory accessed using CoAP over DTLS"

   * Service name: "core-rd", Protocol: "tcp", Description: "Resource
      Directory accessed using CoAP over TCP"

   * Service name "core-rd-tls", Protocol: "tcp", Description:
      "Resource Directory accessed using CoAP over TLS"

   All in common have this document as their reference.

10. Examples

   Two examples are presented: a Lighting Installation example in
   Section 10.1 and a LwM2M example in Section 10.2.

10.1. Lighting Installation

   This example shows a simplified lighting installation which makes use
   of the 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 as described in
   Appendix A. 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.

10.1.1. 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 infrastructure. Therefore, SLAAC
IPv6 addresses are assigned to CT, RD, luminaries and the sensor.
The addresses shown in Table 4 below stand in for these in the
following examples.

```
+=================+================+
| Name            | IPv6 address   |
+=================+================+
| luminary1       | 2001:db8:4::1  |
+-----------------+----------------+
| luminary2       | 2001:db8:4::2  |
+-----------------+----------------+
| Presence sensor | 2001:db8:4::3  |
+-----------------+----------------+
| RD              | 2001:db8:4::ff |
+-----------------+----------------+
```

                    Table 4: Addresses used in the
                              examples

In Section 10.1.2 the use of RD during installation is presented.

10.1.2. 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=tag:example.com,2020:light, and
the presence sensor has rt=tag:example.com,2020: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 RD are
shown in Table 5 below:

```
+=========+===============+========+==============================+
| Name    | endpoint      |resource| resource type                |
|         |               |path    |                              |
+=========+===============+========+==============================+
|luminary1|lm_R2-4-015_wndw|/light/| tag:example.com,2020:light   |
|         |               |left    |                              |
+---------+---------------+--------+------------------------------+
|luminary1|lm_R2-4-015_wndw|/light/| tag:example.com,2020:light   |
|         |               |middle  |                              |
+---------+---------------+--------+------------------------------+
|luminary1|lm_R2-4-015_wndw|/light/| tag:example.com,2020:light   |
|         |               |right   |                              |
+---------+---------------+--------+------------------------------+
|luminary2|lm_R2-4-015_door|/light/| tag:example.com,2020:light   |
|         |               |left    |                              |
+---------+---------------+--------+------------------------------+
|luminary2|lm_R2-4-015_door|/light/| tag:example.com,2020:light   |
|         |               |middle  |                              |
+---------+---------------+--------+------------------------------+
|luminary2|lm_R2-4-015_door|/light/| tag:example.com,2020:light   |
|         |               |right   |                              |
+---------+---------------+--------+------------------------------+
|Presence |ps_R2-4-015_door|/ps    | tag:example.com,2020:p-sensor|
|sensor   |               |        |                              |
+---------+---------------+--------+------------------------------+
```

Table 5: RD identifiers

It is assumed that the CT has performed RD discovery and has received
a response like the one in the Section 4.3 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="tag:example.com,2020:light",
</light/middle>;rt="tag:example.com,2020:light",
</light/right>;rt="tag:example.com,2020: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="tag:example.com,2020:light",
</light/middle>;rt="tag:example.com,2020:light",
</light/right>;rt="tag:example.com,2020: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=R2-4-015
Payload:
</ps>;rt="tag:example.com,2020:p-sensor"

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

       Figure 24: Example of registrations a CT enters into an RD

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, the resources supported by all group members are
published.

```
Req: POST coap://[2001:db8:4::ff]/rd
  ?ep=grp_R2-4-015&et=core.rd-group&base=coap://[ff05::1]
Payload:
</light/left>;rt="tag:example.com,2020:light",
</light/middle>;rt="tag:example.com,2020:light",
</light/right>;rt="tag:example.com,2020:light"

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

Figure 25: Example of a multicast group a CT enters into an RD

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 being configured to join any
group containing lights, searches for candidate groups and joins
them:

```
Req: GET coap://[2001:db8:4::ff]/rd-lookup/ep
  ?d=R2-4-015&et=core.rd-group&rt=light

Res: 2.05 Content
Payload:
</rd/501>;ep=grp_R2-4-015;et=core.rd-group;
         base="coap://[ff05::1]";rt=core.rd-ep
```

Figure 26: Example of a lookup exchange to find suitable
multicast addresses

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

The presence sensor can learn the presence of groups that support
resources with rt=tag:example.com,2020:light in its own sector by
sending the same request, as used by the luminary. The presence
sensor learns the multicast address to use for sending messages to
the luminaries.

10.2. OMA Lightweight M2M (LwM2M)

OMA LwM2M is a profile for device services based on CoAP, providing
interfaces and operations for device management and device service
enablement.

An LwM2M server is an instance of an LwM2M middleware service layer,
containing an RD ([LwM2M] page 36f).

That RD only implements the registration interface, and no lookup is
implemented. Instead, the LwM2M server provides access to the
registered resources, in a similar way to a reverse proxy.

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. 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, Jan Newmarch, Matthias
Kovatsch, Jaime Jimenez and Ted Lemon 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 RD concepts were originally developed.

12. Changelog

changes from -27 to -28

* Security policies / link confidentiality: Point out the RD's
   obligations that follow from such a policy.

* Simple registration: clarify term "regular registration" by
   introducing it along with the reference to Section 5

* Wording fix in first-come-first-remembered

* Wording fixes in RD definition

* Capitalization: Consistently using "registration resource"

changes from -26 to -27

* In general, this addresses the points that were pointed out in
   https://mailarchive.ietf.org/arch/msg/core/xWLomwwhovkU-
   CPGNxnvs40BhaM/ as having "evolved from the review comments being
   discussed in the interim meetings", and the review comments from
   Esko Dijk that were largely entangled in these points.

* Relaxation of the serialization rules for link-format

   The interpretation of RFC6690 used in Appendix B.4 was shown to be
   faulty. Along with a correction, the common implementations of
   link-format were surveyed again and it was found that the only one

that employed the faulty interpretation can still safely be
upgraded. These were removed from the set considered for Limited
Link Format, making the set of valid Limited Link Format documents
larger.

As a consequence, the prescribed serialization of RD output can be
roughly halved in bytes.

There might be additional usage patterns that are possible with
the new set of constraints, but there is insufficient
implementation and deployment experience with them to warrant a
change changes on that front at this point. The specification can
later be extended compatibly to allow these cases and drop the
requirement of Limited Link Format.

* Add Request freshness subsection

   It is now recommended (with security considerations on
   consequences of not doing it) to require ordering of RD
   operations.

   The Echo mechanism (previously suggested in various places but
   never exclusively) is the one prescribed way of getting this
   ordering, making the echo-request-tag reference normative.

* Improved expression about when an RD needs to verify simple
   registration.

   The simple wording missed the authorization part, and did not
   emphasize that this is a per-deployment property.

* Point out the non-atomic properties of paginated access.

* Clarification around impl-info reference.

* Inconsistencies and extraneous quotings removed from examples.

changes from -25 to -26

* Security policies:

   - The First-Come-First-Remembered policy is added as an example
      and a potential default behavior.

   - Clarify that the mapping between endpoint names and subject
      fields is up to a policy that defines reliance on names, and
      give an example.

      - Random EP names: Point that multiple collisions are possible
        but unlikely.

      - Add pointers to policies:

        o RD replication: Point out that policies may limit that.

        o Registration: Reword (ep, d) mapping to a previous
           registration's resource that could have been read as another
           endpoint taking over an existing registration.

      - Clarify that the security policy is a property of the RD the
        any client may need to verify by checking the RD's
        authorization.

      - Clarify how information from an untrusted RD can be verified

      - Remove speculation about how in detail ACE scopes are obtained.

   * Security considerations:

      - Generalize to all current options for security layers usable
        with CoAP (OSCORE was missing as the text predated RFC8613)

      - Relax the previous SHOULD on secure access to SHOULD where
        protection is indicated by security policies (bringing the text
        in line with the -25 changes)

      - Point out that failure to follow the security considerations
        has implications depending on the protection objective
        described with the security policies

      - Shorten amplification mitigation

      - Add note about information in Registration Resource path.

      - Acknowledge that most host discovery operations are not
        secured; mention consequences and mitigation.

   * Abstract, introduction: removed "or disperse networks"

   * RD discovery:

      - Drop the previously stated assumption that RDAO and any DHCP
        options would only be used together with SLAAC and DHCP for
        address configuration, respectivly.

       - Give concrete guidance for address selection based on RFC6724
          when responding to multicasts

       - RDAO:

          o Clarify that it is an option for RAs and not other ND
             messages.

          o Change Lifetime from 16-bit minutes to 32-bit seconds and
             swap it with Reserved (aligning it with RDNSS which it
             shares other properties as well).

       - Point out that clients may need to check RD authorization
          already in last discovery step

    * Registration:

       - Wording around "mostly mandatory" has been improved, conflicts
          clarified and sector default selection adjusted.

    * Simple registration: Rather than coopting POSTs to /.well-known/
       core, a new resource /.well-known/rd is registered. A historical
       note in the text documents the change.

    * Examples:

       - Use example URIs rather than unclear reg names (unless it's
          RFC6690 examples, which were kept for continuity)

       - The LwM2M example was reduced from an outdated explanation of
          the complete LwM2M model to a summary of how RD is used in
          there, with a reference to the current specification.

       - Luminary example: Explain example addresses

       - Luminary example: Drop reference to coap-group mechanism that's
          becoming obsolete, and thus also to RFC7390

       - Multicast addresses in the examples were changed from
          ff35:30:2001:db8::x to ff35:30:2001:db8:f1::8000:x; the 8000 is
          to follow RFC 3307, and the f1 is for consistency with all the
          other example addresses where 2001:db8::/32 is subnetted to
          2001:db8:x::/48 by groups of internally consistent examples.

    * Use case text enhancements

       - Home and building automation: Tie in with RD

- M2M: Move system design paragraph towards the topic of
  reusability.

* Various editorial fixes in response to Gen-ART and IESG reviews.

* Rename 'Full description of the "Endpoint Type" Registration
  Parameter' section to '... RD Parameter'

* Error handling: Place a SHOULD around the likely cases, and make
  the previous "MUST to the best of their capabilities" a "must".

* impl-info: Add note about the type being WIP

* Interaction tables: list CTs as possible initiators where
  applicable

* Registration update: Relax requirement to not send parameters
  needlessly

* Terminology: Clarify that the CTs' installation events can occur
  multiple times.

* Promote RFCs 7252, 7230 and 8288 to normative references

* Moved Christian Amsuess to first author

changes from -24 to -25

* Large rework of section 7 (Security policies)

  Rather than prescribing which data in the RD _is_ authenticated
  (and how), it now describes what applications built on an RD _can_
  choose to authenticate, show possibilities on how to do it and
  outline what it means for clients.

  This addresses Russ' Genart review points on details in the text
  in a rather broad fashion. That is because the discussion on the
  topic inside the WG showed that that text on security has been
  driven more review-by-review than by an architectural plan of the
  authors and WG.

* Add concrete suggestions (twice as long as registrant number with
  retries, or UUIDs without) for random endpoint names

* Point out that simple registration can have faked origins,
  RECOMMEND mitigation when applicable and suggest the Echo
  mechanism to implement it.

* Reference existing and upcoming specifications for DDOS mitigation
  in CoAP.

* Explain the provenance of the example's multicast address.

* Make "SHOULD" of not manipulating foreign registrations a "should"
  and explain how it is enforced

* Clarify application of RFC6570 to search parameters

* Syntactic fixes in examples

* IANA:

  - Don't announce expected number of registrations (goes to write-
    up)

  - Include syntax as part of a field's validity in entry
    requirements

* Editorial changes

  - Align wording between abstract and introduction

  - Abbreviation normalization: "ER model", "RD"

  - RFC8174 boilerplate update

  - Minor clarity fixes

  - Markup and layouting

changes from -23 to -24

* Discovery using DNS-SD added again

* Minimum lifetime (lt) reduced from 60 to 1

* References added

* IANA considerations

  - added about .well-known/core resource

  - added DNS-SD service names

  - made RDAO option number a suggestion

          - added "reference" field to endpoint type registry

     * Lookup: mention that anchor is a legitimate lookup attribute

     * Terminology and example fixes

     * Layout fixes, esp. the use of non-ASCII characters in figures

     changes from -22 to -23

     * Explain that updates can not remove attributes

     * Typo fixes

     changes from -21 to -22

     * Request a dedicated IPv4 address from IANA (rather than sharing
       with All CoAP nodes)

     * Fix erroneous examples

     * Editorial changes

          - Add figure numbers to examples

          - Update RD parameters table to reflect changes of earlier
            versions in the text

          - Typos and minor wording

     changes from -20 to -21

     (Processing comments during WGLC)

     * Defer outdated description of using DNS-SD to find an RD to the
       defining document

     * Describe operational conditions in automation example

     * Recommend particular discovery mechanisms for some managed network
       scenarios

     changes from -19 to -20

     (Processing comments from the WG chair review)

     * Define the permissible characters in endpoint and sector names

* Express requirements on NAT situations in more abstract terms

* Shifted heading levels to have the interfaces on the same level

* Group instructions for error handling into general section

* Simple Registration: process reflowed into items list

* Updated introduction to reflect state of CoRE in general,
  reference RFC7228 (defining "constrained") and use "IoT" term in
  addition to "M2M"

* Update acknowledgements

* Assorted editorial changes

  - Unify examples style

  - Terminology: RDAO defined and not only expanded

  - Add CT to Figure 1

  - Consistency in the use of the term "Content Format"

changes from -18 to -19

* link-local addresses: allow but prescribe split-horizon fashion
  when used, disallow zone identifiers

* Remove informative references to documents not mentioned any more

changes from -17 to -18

* Rather than re-specifying link format (Modernized Link Format),
  describe a Limited Link Format that's the uncontested subset of
  Link Format

* Acknowledging the -17 version as part of the draft

* Move "Read endpoint links" operation to future specification like
  PATCH

* Demote links-json to an informative reference, and removed them
  from exchange examples

* Add note on unusability of link-local IP addresses, and describe
  mitigation.

   * Reshuffling of sections: Move additional operations and endpoint
      lookup back from appendix, and groups into one

   * Lookup interface tightened to not imply applicability for non
      link-format lookups (as those can have vastly different views on
      link cardinality)

   * Simple registration: Change sequence of GET and POST-response,
      ensuring unsuccessful registrations are reported as such, and
      suggest how devices that would have required the inverse behavior
      can still cope with it.

   * Abstract and introduction reworded to avoid the impression that
      resources are stored in full in the RD

   * Simplify the rules governing when a registration resource can or
      must be changed.

   * Drop a figure that has become useless due to the changes of and
      -13 and -17

   * Wording consistency fixes: Use "Registrations" and "target
      attributes"

   * Fix incorrect use of content negotiation in discovery interface
      description (Content-Format -> Accept)

   * State that the base attribute value is part of endpoint lookup
      even when implicit in the registration

   * Update references from RFC5988 to its update RFC8288

   * Remove appendix on protocol-negotiation (which had a note to be
      removed before publication)

   changes from -16 to -17

   (Note that -17 is published as a direct follow-up to -16, containing
   a single change to be discussed at IETF103)

   * Removed groups that are enumerations of registrations and have
      dedicated mechanism

   * Add groups that are enumerations of shared resources and are a
      special case of endpoint registrations

   changes from -15 to -16

* Recommend a common set of resources for members of a group

* Clarified use of multicast group in lighting example

* Add note on concurrent registrations from one EP being possible
  but not expected

* Refresh web examples appendix to reflect current use of Modernized
  Link Format

* Add examples of URIs where Modernized Link Format matters

* Editorial changes

changes from -14 to -15

* Rewrite of section "Security policies"

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

* Renamed "Registree-EP" to Registrant-EP"

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

* Fixed examples

* Editorial changes

changes from -13 to -14

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

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

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

* 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

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

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

* Lookup: lt are not exposed any more

* con/base: Paths are allowed

* Registration resource locations can not have query or fragment
   parts

* Default life time extended to 25 hours

* clarified registration update rules

* lt-value semantics for lookup clarified.

* added template for simple registration

changes from -12 to -13

* Added "all resource directory" nodes MC address

* Clarified observation behavior

* version identification

* example rt= and et= values

* domain from figure 2

* more explanatory text

* endpoints of a groups hosted by different RD

* 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

   *   added Content Model section, including ER diagram

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

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

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

   *   updated LWM2M description

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

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

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

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

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

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

   *   enhanced IANA registry text

   *   state that lookup resources can be observable

   *   More examples and improved text

   changes from -09 to -10

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

   *   removed all text concerning DNS-SD.

   *   removed inconsistency in RDAO text.

   *   suggestions taken over from various sources

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

   * moved simple registration to registration section

   changes from -08 to -09

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

   * changed "ins" ABNF notation.

   * various editorial improvements, including in examples

   * clarifications for RDAO

   changes from -07 to -08

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

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

   * 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

   * add IPv6 ND Option for discovery of an RD

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

   * removed all superfluous client-server diagrams

   * simplified lighting example

   * introduced Commissioning Tool

   * RD-Look-up text is extended.

   changes from -06 to -07

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

   * editorial updates to section 9

   * update author information

   * minor text corrections

Changes from -05 to -06

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

changes from -04 to -05

*   added Update Endpoint Links using PATCH

*   http access made explicit in interface specification

*   Added http examples

Changes from -03 to -04:

*   Added http response codes

*   Clarified endpoint name usage

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

Changes from -02 to -03:

*   Added an example for lighting and DNS integration

*   Added an example for RD use in OMA LWM2M

*   Added Read Links operation for link inspection by endpoints

*   Expanded DNS-SD section

*   Added draft authors Peter van der Stok and Michael Koster

Changes from -01 to -02:

*   Added a catalogue use case.

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

*   Additional examples section added for more complex use cases.

*   New DNS-SD mapping section.

*   Added text on endpoint identification and authentication.

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

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

   Changes from -00 to -01:

   * Removed the ETag validation feature.

   * Place holder for the DNS-SD mapping section.

   * Explicitly disabled GET or POST on returned Location.

   * New registry for RD parameters.

   * Added support for the JSON Link Format.

   * Added reference to the Groupcomm WG draft.

   Changes from -05 to WG Document -00:

   * Updated the version and date.

   Changes from -04 to -05:

   * Restricted Update to parameter updates.

   * Added pagination support for the Lookup interface.

   * Minor editing, bug fixes and reference updates.

   * Added group support.

   * Changed rt to et for the registration and update interface.

   Changes from -03 to -04:

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

   * Integrated the Simple Directory Discovery from Carsten.

   * Editorial improvements.

   * Fixed the use of ETags.

   * Fixed tickets 383 and 372

   Changes from -02 to -03:

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

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

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

   * Improved the security considerations section.

   * Made the POST registration interface idempotent by requiring the
      ep= parameter to be present.

   Changes from -01 to -02:

   * Added a terminology section.

   * Changed the inclusion of an ETag in registration or update to a
      MAY.

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

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

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

13. References

13.1. Normative References

   [I-D.ietf-core-echo-request-tag]
              Amsüss, C., Mattsson, J. P., and G. Selander, "CoAP: Echo,
              Request-Tag, and Token Processing", Work in Progress,
              Internet-Draft, draft-ietf-core-echo-request-tag-12, 1
              February 2021, <https://www.ietf.org/archive/id/draft-
              ietf-core-echo-request-tag-12.txt>.

   [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>.

   [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>.

   [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>.

   [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>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

13.2. Informative References

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

   [I-D.bormann-t2trg-rel-impl]
              Bormann, C., "impl-info: A link relation type for
              disclosing implementation information", Work in Progress,
              Internet-Draft, draft-bormann-t2trg-rel-impl-02, 27
              September 2020, <https://www.ietf.org/archive/id/draft-
              bormann-t2trg-rel-impl-02.txt>.

   [I-D.hartke-t2trg-coral]
              Hartke, K., "The Constrained RESTful Application Language
              (CoRAL)", Work in Progress, Internet-Draft, draft-hartke-
              t2trg-coral-09, 8 July 2019,
              <https://www.ietf.org/archive/id/draft-hartke-t2trg-coral-
              09.txt>.

   [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)", Work in Progress, Internet-Draft,
              draft-ietf-ace-oauth-authz-37, 4 February 2021,
              <https://www.ietf.org/archive/id/draft-ietf-ace-oauth-
              authz-37.txt>.

   [I-D.ietf-core-links-json]
              LI, K., Rahman, A., and C. Bormann, "Representing
              Constrained RESTful Environments (CoRE) Link Format in
              JSON and CBOR", Work in Progress, Internet-Draft, draft-
              ietf-core-links-json-10, 26 February 2018,
              <https://www.ietf.org/archive/id/draft-ietf-core-links-
              json-10.txt>.

   [I-D.ietf-core-rd-dns-sd]
              Stok, P. V. D., Koster, M., and C. Amsüss, "CoRE Resource
              Directory: DNS-SD mapping", Work in Progress, Internet-
              Draft, draft-ietf-core-rd-dns-sd-05, 7 July 2019,
              <https://www.ietf.org/archive/id/draft-ietf-core-rd-dns-
              sd-05.txt>.

   [I-D.silverajan-core-coap-protocol-negotiation]
              Silverajan, B. and M. Ocak, "CoAP Protocol Negotiation",
              Work in Progress, Internet-Draft, draft-silverajan-core-
              coap-protocol-negotiation-09, 2 July 2018,
              <https://www.ietf.org/archive/id/draft-silverajan-core-
              coap-protocol-negotiation-09.txt>.

   [LwM2M]   Open Mobile Alliance, "Lightweight Machine to Machine
              Technical Specification: Transport Bindings (Candidate
              Version 1.1)", 12 June 2018,

            <https://openmobilealliance.org/RELEASE/LightweightM2M/
            V1_1-20180612-C/OMA-TS-LightweightM2M_Transport-
            V1_1-20180612-C.pdf>.

   [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6
            Multicast Addresses", RFC 3306, DOI 10.17487/RFC3306,
            August 2002, <https://www.rfc-editor.org/info/rfc3306>.

   [RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix
            Reserved for Documentation", RFC 3849,
            DOI 10.17487/RFC3849, July 2004,
            <https://www.rfc-editor.org/info/rfc3849>.

   [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
            Unique IDentifier (UUID) URN Namespace", RFC 4122,
            DOI 10.17487/RFC4122, July 2005,
            <https://www.rfc-editor.org/info/rfc4122>.

   [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,
            "Transmission of IPv6 Packets over IEEE 802.15.4
            Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007,
            <https://www.rfc-editor.org/info/rfc4944>.

   [RFC5771] Cotton, M., Vegoda, L., and D. Meyer, "IANA Guidelines for
            IPv4 Multicast Address Assignments", BCP 51, RFC 5771,
            DOI 10.17487/RFC5771, March 2010,
            <https://www.rfc-editor.org/info/rfc5771>.

   [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
            "Default Address Selection for Internet Protocol Version 6
            (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
            <https://www.rfc-editor.org/info/rfc6724>.

   [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>.

   [RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing
            IPv6 Zone Identifiers in Address Literals and Uniform
            Resource Identifiers", RFC 6874, DOI 10.17487/RFC6874,
            February 2013, <https://www.rfc-editor.org/info/rfc6874>.

   [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for
            Constrained-Node Networks", RFC 7228,
            DOI 10.17487/RFC7228, May 2014,
            <https://www.rfc-editor.org/info/rfc7228>.

   [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>.

   [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli,
              "IPv6 Router Advertisement Options for DNS Configuration",
              RFC 8106, DOI 10.17487/RFC8106, March 2017,
              <https://www.rfc-editor.org/info/rfc8106>.

   [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>.

   [RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names
              (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017,
              <https://www.rfc-editor.org/info/rfc8141>.

   [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz,
              "Object Security for Constrained RESTful Environments
              (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019,
              <https://www.rfc-editor.org/info/rfc8613>.

Appendix A. Groups Registration and Lookup

   The RD-Groups usage pattern allows announcing application groups
   inside an RD.

   Groups are represented by endpoint registrations. Their base address
   is a multicast address, and they SHOULD be entered with the endpoint
   type "core.rd-group". The endpoint name can also be referred to as a
   group name in this context.

   The registration is inserted into the RD by a Commissioning Tool,
   which might also be known as a group manager here. It performs third
   party registration and registration updates.

   The links it registers SHOULD be available on all members that join
   the group. Depending on the application, members that lack some
   resource MAY be permissible if requests to them fail gracefully.

   The following example shows a CT registering a group with the name
   "lights" which provides two resources. The directory resource path
   /rd is an example RD location discovered in a request similar to
   Figure 5. The group address in the example is constructed from
   [RFC3849]'s reserved 2001:db8:: prefix as a unicast-prefix based
   site-local address (see [RFC3306].

```
Req: POST coap://rd.example.com/rd?ep=lights&et=core.rd-group
                            &base=coap://[ff35:30:2001:db8:f1::8000:1]
Content-Format: 40
Payload:
</light>;rt="tag:example.com,2020:light";
     if="tag:example.net,2020:actuator",
</color-temperature>;if="tag:example.net,2020:parameter";u=K

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

               Figure 27: Example registration of a group

In this example, the group manager can easily permit devices that
have no writable color-temperature to join, as they would still
respond to brightness changing commands. Had the group instead
contained a single resource that sets brightness and color
temperature atomically, endpoints would need to support both
properties.

The resources of a group can be looked up like any other resource,
and the group registrations (along with any additional registration
parameters) can be looked up using the endpoint lookup interface.

The following example shows a client performing an endpoint lookup
for all groups.

```
Req: GET /rd-lookup/ep?et=core.rd-group

Res: 2.05 Content
Payload:
</rd/12>;ep=lights&et=core.rd-group;
        base="coap://[ff35:30:2001:f1:db8::8000:1]";rt=core.rd-ep
```

                   Figure 28: Example lookup of groups

The following example shows a client performing a lookup of all
resources of all endpoints (groups) with et=core.rd-group.

```
Req: GET /rd-lookup/res?et=core.rd-group

Res: 2.05 Content
Payload:
<coap://[ff35:30:2001:db8:f1::8000:1]/light>;
     rt="tag:example.com,2020:light";
     if="tag:example.net,2020:actuator",
<coap://[ff35:30:2001:db8:f1::8000:1]/color-temperature>;
     if="tag:example.net,2020:parameter";u=K,
```

                Figure 29: Example lookup of resources inside groups

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 header fields, 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:

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

   Res: 2.05 Content
   Payload:
   </sensors/temp>;rt=temperature;ct=0

                 Figure 30: Example of direct resource discovery

   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: "sensors" and "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 ""/sensors/temp"", which is a relative
   URI that needs resolving. The base URI
   <coap://[ff02::fd]:5683/.well-known/core> is used to resolve the
   reference /sensors/temp against.

The Base URI of the requested resource can be composed from the
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"".

Because ""/sensors/temp"" starts with a single slash, 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
""/sensors/temp"" into ""coap://[2001:db8:f0::1]/sensors/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 format 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 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 implied to be
"coap:://[2001:db8:f0::1]" by the default value of the anchor (see
Appendix B.4). A full English expression of the "host relation" is:

'"coap://[2001:db8:f0::1]" is hosting the resource
"coap://[2001:db8:f0::1]/sensors/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:

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

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

        Figure 31: Extended example of direct resource discovery

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 RD 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 RD that was announced to
it, sending this request from its UDP port "[2001:db8:f0::1]:6553":

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

Res: 2.04 Changed

                Figure 32: Example of a simple registration

The RD 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 ask it for all temperature resources:

```
Req: GET coap://[2001:db8:f0::ff]/rd-lookup/res?rt=temperature

Res: 2.05 Content
Payload:
<coap://[2001:db8:f0::1]/sensors/temp>;rt=temperature;ct=0
```

Figure 33: Example exchange performing resource lookup

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]/sensors/temp", which is of the resource type "temperature" and can be accessed using the text/plain content format.'

To complete the examples, the client could also query all resources hosted at the endpoint with the known endpoint name "simple-host1":

```
Req: GET coap://[2001:db8:f0::ff]/rd-lookup/res?ep=simple-host1

Res: 2.05 Content
Payload:
<coap://[2001:db8:f0::1]/sensors/temp>;rt=temperature;ct=0,
<coap://[2001:db8:f0::1]/sensors/light>;rt=light-lux;ct=0,
<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
```

Figure 34: Extended example exchange performing resource lookup

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 done an equivalent full registration 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/sensors/temp>;rt=temperature;ct=0
```

Figure 35: Example payload of a response to a resource lookup
                    with a dedicated base URI

and analogous records.

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

   While link-format and Link header fields look very similar and are
   based on the same model of typed links, there are some differences
   between [RFC6690] and [RFC8288]. When implementing an RD or
   interacting with an RD, care must be taken to follow the [RFC6690]
   behavior whenever application/link-format representations are used.

   * "Default value of anchor": Both under [RFC6690] and [RFC8288],
      relative references in the term inside the angle brackets (the
      target) and the anchor attribute are resolved against the relevant
      base URI (which usually is the URI used to retrieve the entity),
      and independent of each other.

      When, in an [RFC8288] Link header, the anchor attribute is absent,
      the link's context is the URI of the selected representation (and
      usually equal to the base URI).

      In [RFC6690] links, if the anchor attribute is absent, the default
      value is the Origin of (for all relevant cases: the URI reference
      "/" resolved against) the link's target.

   * 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 header
      fields are practically ASCII strings that use percent encoding for
      non-ASCII characters, stating the encoding explicitly when
      required.

      For example, while a Link header field 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/Malmö>;rel=live-environment-data

Appendix C. Limited Link Format

   The CoRE Link Format as described in [RFC6690] has been interpreted
   differently by implementers, and a strict implementation rules out
   some use cases of an RD (e.g. base values with path components in
   combination with absent anchors).

This appendix describes a subset of link format documents called
Limited Link Format. The one rule herein is not very limiting in
practice -- all examples in RFC6690, and all deployments the authors
are aware of already stick to them -- but ease the implementation of
RD servers.

It is applicable to representations in the application/link-format
media type, and any other media types that inherit [RFC6690]
Section 2.1.

A link format representation is in Limited Link format if, for each
link in it, the following applies:

All URI references either follow the URI or the path-absolute ABNF
rule of RFC3986 (i.e. target and anchor each either start with a
scheme or with a single slash).

Authors' Addresses

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

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

Zach Shelby
ARM
150 Rose Orchard
San Jose, 95134
United States of America

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

Michael Koster
SmartThings
665 Clyde Avenue
Mountain View, 94043
United States of America

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

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28359 Bremen
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