CoRE
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

Z. Shelby
ARM
M. Koster
SmartThings
C. Bormann
Universitaet Bremen TZI
P. van der Stok
consultant

March 13, 2017

CoRE Resource Directory
draft-ietf-core-resource-directory-10

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The work on Constrained RESTful Environments (CoRE) aims at realizing the REST architecture in a suitable form for the most constrained nodes (e.g., 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN). CoRE is aimed at machine-to-machine (M2M) applications such as smart energy and building automation.

The discovery of resources offered by a constrained server is very important in machine-to-machine applications where there are no humans in the loop and static interfaces result in fragility. The discovery of resources provided by an HTTP Web Server is typically called Web Linking [RFC5988]. The use of Web Linking for the description and discovery of resources hosted by constrained web servers is specified by the CoRE Link Format [RFC6690]. However, [RFC6690] only describes how to discover resources from the web server that hosts them by requesting "/.well-known/core". In many M2M scenarios, direct discovery of resources is not practical due to sleeping nodes, disperse networks, or networks where multicast traffic is inefficient. These problems can be solved by employing an entity called a Resource Directory (RD), which hosts descriptions of resources held on other servers, allowing lookups to be performed for those resources.

This document specifies the web interfaces that a Resource Directory supports in order for web servers to discover the RD and to register, maintain, lookup and remove resource descriptions. Furthermore, new link attributes useful in conjunction with a Resource Directory are defined. Although the examples in this document show the use of

these interfaces with CoAP [RFC7252], they can be applied in an equivalent manner to HTTP [RFC7230].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. The term "byte" is used in its now customary sense as a synonym for "octet".

This specification requires readers to be familiar with all the terms and concepts that are discussed in [RFC5988] and [RFC6690]. Readers should also be familiar with the terms and concepts discussed in [RFC7252]. To describe the REST interfaces defined in this specification, the URI Template format is used [RFC6570].

This specification makes use of the following additional terminology:

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

Domain
: In the context of a Resource Directory, a domain is a logical grouping of endpoints. This specification assumes that the list of Domains supported by an RD is pre-configured by that RD. When a domain is exported to DNS, the domain value equates to the DNS domain name.

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

Endpoint
: Endpoint (EP) is a term used to describe a web server or client in [RFC7252]. In the context of this specification an endpoint is used to describe a web server that registers resources to the Resource Directory. An endpoint is identified by its endpoint name, which is included during registration, and is unique within the associated domain of the registration.

Context
: When registering links to a Resource Directory, the Context refers to the scheme, address, port, and base path for all the links registered on behalf of an endpoint, of the general form

scheme://host:port/path/ where the client may explicitly set the scheme and host, and may supply the port and path as optional parameters. When the context of a registration is explicitly set, the URI resolution rules in [RFC3986] MUST be applied.

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

RDAO
Resource Directory Address Option.

3. Architecture and Use Cases

The resource directory architecture is illustrated in Figure 1. A Resource Directory (RD) is used as a repository for Web Links [RFC5988] about resources hosted on other web servers, which are called endpoints (EP). An endpoint is a web server associated with a scheme, IP address and port (called Context), thus a physical node may host one or more endpoints. The RD implements a set of REST interfaces for endpoints to register and maintain sets of Web Links (called resource directory registration entries), and for clients to lookup resources from the RD or maintain groups. Endpoints themselves can also act as clients. An RD can be logically segmented by the use of Domains. The domain an endpoint is associated with can be defined by the RD or configured by an outside entity. This information hierarchy is shown in Figure 2.

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

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

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

At the first registration of a set of entries, a "registration resource" is created, the location of which is returned to the registering endpoint. The registering endpoint uses this registration resource to manage the contents of the registration entry.

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

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

Figure 1: The resource directory architecture.

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

Figure 2: The resource directory information hierarchy.

3.1. Use Case: Cellular M2M

Over the last few years, mobile operators around the world have focused on development of M2M solutions in order to expand the business to the new type of users: machines. The machines are connected directly to a mobile network using an appropriate embedded air interface (GSM/GPRS, WCDMA, LTE) or via a gateway providing short and wide range wireless interfaces. From the system design point of

view, the ambition is to design horizontal solutions that can enable utilization of machines in different applications depending on their current availability and capabilities as well as application requirements, thus avoiding silo like solutions. One of the crucial enablers of such design is the ability to discover resources (machines -- endpoints) capable of providing required information at a given time or acting on instructions from the end users.

In a typical scenario, during a boot-up procedure (and periodically afterwards), the machines (endpoints) register with a Resource Directory (for example EPs installed on vehicles enabling tracking of their position for fleet management purposes and monitoring environment parameters) hosted by the mobile operator or somewhere else in the network, periodically a description of its own capabilities. Due to the usual network configuration of mobile networks, the EPs attached to the mobile network may not always be efficiently reachable. Therefore, a remote server is usually used to provide proxy access to the EPs. The address of each (proxy) endpoint on this server is included in the resource description stored in the RD. The users, for example mobile applications for environment monitoring, contact the RD, look up the endpoints capable of providing information about the environment using appropriate set of link parameters, obtain information on how to contact them (URLs of the proxy server) and then initiate interaction to obtain information that is finally processed, displayed on the screen and usually stored in a database. Similarly, fleet management systems provide the appropriate link parameters to the RD to look up for EPs deployed on the vehicles the application is responsible for.

3.2. Use Case: Home and Building Automation

Home and commercial building automation systems can benefit from the use of M2M web services. The discovery requirements of these applications are demanding. Home automation usually relies on run-time discovery to commission the system, whereas in building automation a combination of professional commissioning and run-time discovery is used. Both home and building automation involve peer-to-peer interactions between endpoints, and involve battery-powered sleeping devices.

3.3. Use Case: Link Catalogues

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

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

Metadata in web link formats like [RFC6690] are supplied by Resource Directories, which may be internally stored as triples, or relation/attribute pairs providing metadata about resource links. External catalogs that are represented in other formats may be converted to common web linking formats for storage and access by Resource Directories. Since it is common practice for these to be URN encoded, simple and lossless structural transforms should generally be sufficient to store external metadata in Resource Directories.

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

4. Finding a Resource Directory

Several mechanisms can be employed for discovering the RD, including assuming a default location (e.g. on an Edge Router in a LoWPAN), assigning an anycast address to the RD, using DHCP, or discovering the RD using .well-known/core and hyperlinks as specified in CoRE Link Format [RFC6690]. Endpoints that want to contact a Resource Directory can obtain candidate IP addresses for such servers in a number of ways.

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

- o specific static configuration (e.g., anycast addresses), if any,
- o the ABRO option of 6LoWPAN-ND [RFC6775],
- o other ND options that happen to point to servers (such as RDNSS),
- o DHCPv6 options that might be defined later.
- o The IPv6 Neighbor Discovery Resource Directory Address Option described in Section 4.1

In networks with more inexpensive use of multicast, the candidate IP address may be a well-known multicast address, i.e. directory servers are found by simply sending GET requests to that well-known multicast address (see Section 5.2).

Constrained nodes configured in large batches may be configured for an anycast address for the RD. 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 RD that is appropriate for the environment.

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

4.1. Resource Directory Address Option (RDAO)

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

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

The lifetime 0x0 means that the RD address is invalid and to be removed.

The RDAO format is:

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

Fields:

Type:	38
Length:	8-bit unsigned integer. The length of the option in units of 8 bytes. Always 3.
Valid Lifetime:	16-bit unsigned integer. The length of time in units of 60 seconds (relative to the time the packet is received) that this Resource Directory address is valid. A value of all zero bits (0x0) indicates that this Resource Directory address is not valid anymore.
Reserved:	This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
RD Address:	IPv6 address of the RD.

Figure 3: Resource Directory Address Option

5. Resource Directory

This section defines the required set of REST interfaces between a Resource Directory (RD) and endpoints. Although the examples throughout this section assume the use of CoAP [RFC7252], these REST interfaces can also be realized using HTTP [RFC7230]. In all definitions in this section, both CoAP response codes (with dot notation) and HTTP response codes (without dot notation) are shown.

An RD implementing this specification MUST support the discovery, registration, update, lookup, and removal interfaces defined in this section.

5.1. Content Formats

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

Resource Directories implementing this specification MAY support additional content formats.

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

5.2. Base URI Discovery

Before an endpoint can make use of an RD, it must first know the RD's address and port, and the base URI information for its REST API. This section defines discovery of the RD and its base URI using the well-known interface of the CoRE Link Format [RFC6690]. It is however expected that RDs will also be discoverable via other methods depending on the deployment.

Discovery of the RD base 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 base URI for RD Lookup operations, and "core.gp" is used to discover the base URI for RD Group operations. Upon success, the response will contain a payload with a link format entry for each RD function discovered, indicating the base 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.

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

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

points SHOULD be provided to enable the bootstrapping of unicast discovery.

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

The discovery request interface is specified as follows:

Interaction: EP -> RD

Method: GET

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

URI Template Variables:

rt := Resource Type (optional). MAY contain one of the values "core.rd", "core.rd-lookup*", "core.rd-lookup-d", "core.rd-lookup-res", "core.rd-lookup-ep", "core.rd-lookup-gp", "core.rd-group" or "core.rd*"

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

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

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

The following response codes are defined for this interface:

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

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

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

Failure: No error response to a multicast request.

HTTP support : YES (Unicast only)

The following example shows an endpoint discovering an RD using this interface, thus learning that the base RD resource is, in this example, at /rd 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 base RD resource, although diagnostics and debugging is facilitated by using the base paths specified here where possible.

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

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

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 more application-specific content format (picked as 65225 in this example; this is in the experimental space, not an assigned value). The base RD resource values /rd, /rd-lookup, and /rd-group are example values.

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

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

5.3. Registration

After discovering the location of an RD, an endpoint MAY register its resources using the registration interface. This interface accepts a POST from an endpoint containing the list of resources to be added to the directory as the message payload in the CoRE Link Format [RFC6690], JSON CoRE Link Format (application/link-format+json), or CBOR CoRE Link Format (application/link-format+cbor) [I-D.ietf-core-links-json], along with query parameters indicating the name of the endpoint, and optionally its domain and the lifetime 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. An endpoint MUST use that location when refreshing registrations using this interface. Endpoint resources in the RD are kept active for the period indicated by the lifetime parameter. The endpoint is responsible for refreshing the entry within this period using either

the registration or update interface. The registration interface MUST be implemented to be idempotent, so that registering twice with the same endpoint parameters ep and d does not create multiple RD entries. A new registration may be created at any time to supersede an existing registration, replacing the registration parameters and links.

The registration request interface is specified as follows:

Interaction: EP -> RD

Method: POST

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

URI Template Variables:

rd := RD Base URI path (mandatory). This is the path of the RD, as obtained from discovery. The value "rd" is recommended for this variable.

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

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

et := Endpoint Type (optional). The semantic type of the endpoint. This parameter SHOULD be less than 63 bytes.

lt := Lifetime (optional). Lifetime of the registration in seconds. Range of 60-4294967295. If no lifetime is included in the initial registration, a default value of 86400 (24 hours) SHOULD be assumed. If the lt parameter is not included in a registration refresh or update operation, the most recently supplied value SHALL be re-used.

con := Context (optional). This parameter sets the scheme, address and port at which this server is available in the form scheme://host:port/path. In the absence of this parameter the scheme of the protocol, source address and source port of the register request are assumed. This parameter is mandatory when the directory is filled by a third party such as an commissioning tool. When con is used, scheme and host are mandatory and port and path parameters are optional. If the

endpoint uses an ephemeral port to register with, it MUST include the con: parameter in the registration to provide a valid network path. If the endpoint which is located behind a NAT gateway is registering with a Resource Directory which is on the network service side of the NAT gateway, the endpoint MUST use a persistent port for the outgoing registration in order to provide the NAT gateway with a valid network address for replies and incoming requests.

Content-Format: application/link-format

Content-Format: application/link-format+json

Content-Format: application/link-format+cbor

The following response codes are defined for this interface:

Success: 2.01 "Created" or 201 "Created". The Location header option MUST be included in the response when a new registration resource is created. 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.

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

Failure: 4.09 "Conflict" or 409 "Conflict". Attempt to update the registration content with links resulting in plurality of references; see Section 5.3.4.

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

HTTP support: YES

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

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

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

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

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

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

5.3.1. **Simple Registration**

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

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

The endpoint then finds one or more addresses of the directory server as described in Section 4.

An endpoint can send (a selection of) hosted resources to a directory server for publication as described in Section 5.3.2.

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

5.3.2. Simple publishing to Resource Directory Server

An endpoint that wants to make itself discoverable occasionally sends a POST request to the "/.well-known/core" URI of any candidate directory server that it finds. The body of the POST request is empty, which triggers the resource directory server to perform GET requests at the requesting server's default discovery URI to obtain the link-format payload to register.

The endpoint MUST include the endpoint name and MAY include the registration parameters d, lt, and et, in the POST request as per Section 5.3.

The following example shows an endpoint using simple publishing, by simply sending an empty POST to a resource directory.

```
Req:(to RD server from [ff02::1])
POST coap://rd.example.com/.well-known/core?lt=6000;ep=node1

Content-Format: 40

payload:

(empty payload)

Res: 2.04 Changed

(later)

Req: (from RD server to [ff02::1])
GET coap://[ff02::1]/.well-known/core

Accept: 40

Res: 2.05 Content

payload:

</sen/temp>
```

5.3.3. Third-party registration

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

In a controlled environment (e.g. building control), the Resource Directory can be filled by a third device, called a commissioning

tool. The commissioning tool can fill the Resource Directory from a database or other means. For that purpose the scheme, IP address and port of the registered device is indicated in the Context parameter of the registration described in Section 5.3.

5.3.4. Plurality of link references in a Registration

Plurality of link references within a Registration (registration resource) is an indication of some error condition and should not be allowed.

Plurality of link references exists if, and only if, two or more links in a Registration contain identical context, target, and relation values. This condition would be likely to arise if there were multiple co-ordinators or configuration tools, each with a different set of configuration values for the same resource.

A Resource Directory SHOULD reject a registration, or an operation on a registration, which would result in a plurality of link references within the the context of the registration. There is no requirement in this document for a resource directory to check for plurality of reference between different registrations. Resource Directory operations which are rejected due to reference plurality SHOULD be returned the "Conflict" code, indicating that there is someting wrong with the request.

5.4. Operations on the Registration Resource

After the initial registration, an endpoint should retain the returned location of the Registration Resource for further operations, including refreshing the registration in order to extend the lifetie and "keep-alive" the registration. If the lifetime of the registration expires, the RD SHOULD NOT respond to discovery queries with information from the 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 resource recovery and garbage collection. If the Registration Resource is removed, the endpoint will need to re-register.

The Registration Resource may also be used to inspect the registration resource using GET, update the registration link contents using PATCH, or cancel the registration using DELETE.

These operations are described in this section.

In accordance with Section 5.3.4, operations which would result in plural link references within the context of a registration resource SHOULD be rejected using the "Conflict" result code.

5.4.1. Registration Update

The update interface is used by an endpoint to refresh or update its registration with an RD. To use the interface, the endpoint sends a POST request to the registration resource returned in the Location header option in the response returned from the intial registration operation.

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

Upon receiving an update request, an RD MUST reset the timeout for that endpoint and update the scheme, IP address and port of the endpoint, using the source address of the update, or the context ("con") parameter if present. If the lifetime parameter "lt" is included in the received update request, the RD MUST update the lifetime of the registration and set the timeout equal to the new lifetime. If the lifetime parameter is not included in the registration update, the most recent setting is re-used for the next registration time-out period.

An update MAY optionally add or replace links for the endpoint by including those links in the payload of the update as a CoRE Link Format document. A link is replaced only if all of the target URI and relation type (if present) and anchor value (if present) match.

If the link payload is included, it SHOULD be checked for reference plurality as described in Section 5.3.4 and rejected with a "Conflict" result if there are plural link references detected.

In addition to the use of POST, as described in this section, there is an alternate way to add, replace, and delete links using PATCH as described in Section 5.4.4.

The update registration request interface is specified as follows:

Interaction: EP -> RD

Method: POST

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

URI Template Variables:

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

lt := Lifetime (optional). Lifetime of the registration in seconds. Range of 60-4294967295. If no lifetime is included, the previous last lifetime set on a previous update or the original registration (falling back to 86400) SHOULD be used.

con := Context (optional). This parameter sets the scheme, address and port at which this server is available in the form scheme://host:port/path. In the absence of this parameter the scheme of the protocol, source address and source port of the register request are assumed. This parameter is mandatory when the directory is filled by a third party such as an commissioning tool. When con is used, scheme and host are mandatory and port and path parameters are optional.

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

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

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

The following response codes are defined for this interface:

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

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

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

Failure: 4.09 "Conflict" or 409 "Conflict". Attempt to update the registration content with links resulting in plurality of references; see Section 5.3.4.

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

HTTP support: YES

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

```
Req: POST /rd/4521

Res: 2.04 Changed
```

The following example shows an endpoint updating its registration with a new lifetime and context, changing an existing link, and adding a new link using this interface with the example location value /rd/4521. With the initial registration the client set the following values:

- lifetime (lt)=500

- context (con)=coap://local-proxy-old.example.com:5683

- resource= </sensors/temp>;ct=41;rt="foobar";if="sensor"

```
Req: POST /rd/4521?lt=600&con="coap://local-proxy.example.com:5683"
Content-Format: 40
Payload:
</sensors/temp>;ct=41;rt="temperature-f";if="sensor",
</sensors/door>;ct=41;rt="door";if="sensor"

Res: 2.04 Changed
```

5.4.2. **Registration Removal**

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

The removal request interface is specified as follows:

Interaction: EP -> RD

Method: DELETE

URI Template: /{+location}

URI Template Variables:

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

The following responses codes are defined for this interface:

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

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

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

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

HTTP support: YES

The following examples shows successful removal of the endpoint from the RD with example location value /rd/4521.

```
Req: DELETE /rd/4521

Res: 2.02 Deleted
```

5.4.3. Read Endpoint Links

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

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

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

The read request interface is specified as follows:

Interaction: EP -> RD

Method: GET

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

URI Template Variables:

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

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

The following responses codes are defined for this interface:

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

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

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

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

HTTP support: YES

The following examples show successful read of the endpoint links from the RD, with example location value /rd/4521.

```
Req: GET /rd/4521

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

5.4.4. Update Endpoint Links

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

Other PATCH document formats may be used as appropriate for patching the array of objects format of a Registration Resource. In particular, a select-merge patch document format could combine the function of link selection query and link attribute replacement values.

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

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

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

If no links are selected by the query parameters, the PATCH operation SHOULD NOT update the state of any resource, and SHOULD return a reply of "Changed".

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

If the PATCH payload contains plural link references, or processing the PATCH payload would result in plural link references, the request SHOULD be rejected with a "Conflict" result.

If the PATCH payload results in the modification of link target, context, or relation values, that is "href", "rel", or "anchor", the request SHOULD be rejected with a "Conflict" result code.

The update request interface is specified as follows:

Interaction: EP -> RD

Method: PATCH

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

URI Template Variables:

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

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

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

The following response codes are defined for this interface:

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

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

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

Failure: 4.09 "Conflict" or 409 "Conflict". Attempt to update the registration content with links resulting in plurality of references; see Section 5.3.4.

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

HTTP support: YES

The following examples show an endpoint adding </sensors/humid>, modifying </sensors/temp>, and removing </sensors/light> links in RD using the Update Endpoint Links function with the example location value /rd/4521.

The Registration Resource initial state is:

```
Req: GET /rd/4521

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

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

```
Req: PATCH /rd/4521

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

Content-Format:
application/merge-patch+json

Res: 2.04 Changed
```

```
Req: GET /rd/4521

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

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

```
Req: PATCH /rd/4521?href=/sensors/temp

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

Content-Format:
application/merge-patch+json

Res: 2.04 Changed

Req: GET /rd/4521

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

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

```
Req: PATCH /rd/4521?href=/sensors/light

Payload:
{}

Content-Format:
application/merge-patch+json

Res: 2.04 Changed
```

```
Req: GET /rd/4521

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

6. RD Groups

This section defines the REST API for the creation, management, and lookup of endpoints for group operations. Similar to endpoint registration entries in the RD, groups may be created or removed. However unlike an endpoint entry, a group entry consists of a list of endpoints and does not have a lifetime associated with it. In order to make use of multicast requests with CoAP, a group MAY have a multicast address associated with it.

6.1. Register a Group

In order to create a group, a commissioning tool (CT) used to configure groups, makes a request to the RD indicating the name of the group to create (or update), optionally the domain the group belongs to, and optionally the multicast address of the group. The registration message includes the list of endpoints that belong to that group.

All the endpoints in the group MUST be registered with the RD before registering a group. If an endpoint is not yet registered to the RD before registering the group, the registration message returns an error. The RD sends a blank target URI for every endpoint link when registering the group.

Configuration of the endpoints themselves is out of scope of this specification. Such an interface for managing the group membership of an endpoint has been defined in [RFC7390].

The registration request interface is specified as follows:

Interaction: CT -> RD

Method: POST

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

URI Template Variables:

rd-group := RD Group Base URI path (mandatory). This is the path of the RD Group REST API. The value "rd-group" is recommended for this variable.

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

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

con := Context (optional). This parameter sets the scheme, address and port at which this server is available in the form scheme://host:port/path. In the absence of this parameter the scheme of the protocol, source address and source port of the register request are assumed. This parameter is mandatory when the directory is filled by a third party such as an commissioning tool. When con is used, scheme and host are mandatory and port and path parameters are optional.

Content-Format: application/link-format

Content-Format: application/link-format+json

Content-Format: application/link-format+cbor

The following response codes are defined for this interface:

Success: 2.01 "Created" or 201 "Created". The Location header option MUST be returned in response to a successful group CREATE operation. This Location MUST be a stable identifier generated by the RD as it is used for delete operations of the group registration resource.

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

Failure: 4.04 "Not Found" or 404 "Not Found". An Endpoint is not registered in the RD (e.g. may have expired).

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

HTTP support: YES

The following example shows an EP registering a group with the name "lights" which has two endpoints to an RD using this interface. The base location value /rd-group is an example of an RD base location.

```
Req: POST coap://rd.example.com/rd-group?gp=lights
Content-Format: 40
Payload:
<>;ep="node1",
<>;ep="node2"

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

6.2. Group Removal

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

The removal request interface is specified as follows:

Interaction: CT -> RD

Method: DELETE

URI Template: /{+location}

URI Template Variables:

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

The following responses codes are defined for this interface:

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

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

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

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

HTTP support: YES

The following examples shows successful removal of the group from the RD with the example location value /rd-group/12.

```
Req: DELETE /rd-group/12

Res: 2.02 Deleted
```

7. RD Lookup

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

RD Lookup allows lookups for domains, groups, endpoints and resources using attributes defined in this document and for use with the CoRE Link Format. The result of a lookup request is the list of links (if any) corresponding to the type of lookup. Thus, a domain lookup MUST return a list of domains, a group lookup MUST return a list of groups, an endpoint lookup MUST return a list of endpoints and a resource lookup MUST return a list of links to resources.

RD Lookup does not expose registration resources directly, but returns link content from registration resource entries which satisfy RD Lookup queries.

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
Domain	core.rd-lookup-d	Optional
Group	core.rd-lookup-gp	Optional

Table 1: Lookup Types

Each endpoint and resource lookup result returns respectively the scheme (IP address and port) followed by the path part of the URI of every endpoint and resource inside angle brackets ("<>") and followed by the other parameters.

The target of these links SHOULD be the actual location of the domain, endpoint or resource, but MAY be an intermediate proxy e.g. in the case of an HTTP lookup interface for CoAP endpoints.

The domain lookup returns every lookup domain with a base RD resource value (e.g. "/rd") encapsulated within angle brackets.

In case that a group does not implement any multicast address, the group lookup returns every group lookup with a group base resource value encapsulated within angle brackets (e.g. "/rd/look-up"). Otherwise, the group lookup returns the multicast address of the group inside angle brackets.

Using the Accept Option, the requester can control whether this list is returned in CoRE Link Format ("application/link-format", default) or its alternate content-formats ("application/link-format+json" or "application/link-format+cbor").

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

Multiple query parameters MAY be included in a lookup, all included parameters MUST match for a resource to be returned. The character'*' MAY be included at the end of a parameter value as a wildcard operator.

RD Lookup requests MAY use any set of query parameters to match the registered attributes and relations. In addition, this interface MAY be used with queries that specify domains, endpoints, and groups. For example, a domain lookup filtering on groups would return a list of domains that contain the specified groups. An endpoint lookup filtering on groups would return a list of endpoints that are in the specified groups.

The lookup interface is specified as follows:

Interaction: Client -> RD

Method: GET

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

URI Template Variables:

rd-lookup-base := RD Lookup Base URI path (mandatory). This is the base path for RD Lookup requests. The recommended value for this variable is: "rd-lookup".

lookup-type := ("ep", "res") (mandatory), ("d","gp") (optional) This variable is used to select the type of lookup to perform (endpoint, resource, domain, or group). The values are recommended defaults and MAY use other values as needed. The supported lookup-types SHOULD be listed in .well-known/core using the specified resource types.

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

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

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

gp := Group name (optional). Used for endpoint, group and resource lookups.

page := Page (optional). Parameter can not 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.

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

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

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

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

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

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

The following responses codes are defined for this interface:

Success: 2.05 "Content" or 200 "OK" with an "application/link-format", "application/link-format+cbor", or "application/link-format+json" payload containing matching entries for the lookup.

Failure: 4.04 "Not Found" or 404 "Not Found" in case no matching entry is found for a unicast request.

Failure: No error response to a multicast request.

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

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

HTTP support: YES

The examples in this section assume 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 look-up location /rd-lookup/:

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

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

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

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

Res: 2.05 Content
<coap://[FDFD::127]:61616>;ep="node5",
<coap://[FDFD::129]:61616>;ep="node7"
```

The following example shows a client performing a domain lookup:

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

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

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

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

Res: 2.05 Content
<>;gp="lights1";d="example.com"
<>;gp="lights2";d="ecample.com"
```

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

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

Res: 2.05 Content
<coap://[FDFD::123]:61616>;ep="node1",
<coap://[FDFD::124]:61616>;ep="node2"
```

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

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

Res: 2.05 Content
<>;gp="lights1"
```

The following example shows a client performing a paginated lookup

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

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

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

   Res: 2.05 Content
   <coap://[FDFD::123]:61616/res/5>;rt=sensor;ct=60
   <coap://[FDFD::123]:61616/res/6>;rt=sensor;ct=60
   <coap://[FDFD::123]:61616/res/7>;rt=sensor;ct=60
   <coap://[FDFD::123]:61616/res/8>;rt=sensor;ct=60
   <coap://[FDFD::123]:61616/res/9>;rt=sensor;ct=60
```

8. Security Considerations

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

8.1. Endpoint Identification and Authentication

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

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

8.2. Access Control

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

8.3. Denial of Service Attacks

Services that run over UDP unprotected are vulnerable to unknowingly become part of a DDoS attack as UDP does not require return routability check. Therefore, an attacker can easily spoof the source IP of the target entity and send requests to such a service which would then respond to the target entity. This can be used for large-scale DDoS attacks on the target. Especially, if the service returns a response that is order of magnitudes larger than the request, the situation becomes even worse as now the attack can be amplified. DNS servers have been widely used for DDoS amplification attacks. There is also a danger that NTP Servers could become implicated in denial-of-service (DoS) attacks since they run on unprotected UDP, there is no return routability check, and they can have a large amplification factor. The responses from the NTP server were found to be 19 times larger than the request. A Resource Directory (RD) which responds to wild-card lookups is potentially vulnerable if run with CoAP over UDP. Since there is no return routability check and the responses can be significantly larger than requests, RDs can unknowingly become part of a DDoS amplification attack.

9. IANA Considerations

9.1. Resource Types

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

9.2. IPv6 ND Resource Directory Address Option

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

- o Resource Directory address Option (38)

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 query parameter, validity requirements if any and a description. The query parameter MUST be a valid URI query key [RFC3986].

Initial entries in this sub-registry are as follows:

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

Table 2: RD Parameters

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

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 Resource Directory (RD) with a CoAP interface to facilitate the installation and start up of the application code in the lights and sensors. In particular, the example leads to the definition of a

group and the enabling of the corresponding multicast address. No conclusions must be drawn on the realization of actual installation or naming procedures, because the example only "emphasizes" some of the issues that may influence the use of the RD and does not pretend to be normative. The example uses the recommended values for the base resources: "/rd", "/rd-lookup", and "/rd-group".

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 infra structure. Therefore, SLAAC IPv6 addresses are assigned to CT, RD, luminaries and sensor shown in Table 3 below:

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

Table 3: interface SLAAC addresses

In Section 10.1.2 the use of resource directory 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: light, and the presence sensor has rt: p-sensor. The endpoints have names which are relevant to the light installation manager. In this case luminary1, luminary2, and the presence sensor are located in room 2-4-015, where luminary1 is located at the window and luminary2 and the presence sensor are located at the door. The endpoint names reflect this physical location. The middle, left and right lamps are accessed via path /light/middle, /light/left, and /light/right respectively. The identifiers relevant to the Resource Directory are shown in Table 4 below:

Name	endpoint	resource path	resource type
luminary1	lm_R2-4-015_wndw	/light/left	light
luminary1	lm_R2-4-015_wndw	/light/middle	light
luminary1	lm_R2-4-015_wndw	/light/right	light
luminary2	lm_R2-4-015_door	/light/left	light
luminary2	lm_R2-4-015_door	/light/middle	light
luminary2	lm_R2-4-015_door	/light/right	light
Presence sensor	ps_R2-4-015_door	/ps	p-sensor

Table 4: Resource Directory identifiers

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

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

Res: 2.01 Created
Location: /rd/4521

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

Res: 2.01 Created
Location: /rd/4522

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Dependent on the situation, only the address, "a", or the name, "n", is specified in the coap-group resource.

10.2. OMA Lightweight M2M (LWM2M) Example

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

OMA LWM2M is a profile for device services based on CoAP(OMA Name Authority). LWM2M defines a simple object model and a number of abstract interfaces and operations for device management and device service enablement.

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

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

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

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

The location of the LWM2M Server and RD base 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.

10.2.1. The LWM2M Object Model

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

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

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

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

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

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

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

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

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

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

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

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

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

/1/0/1

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

10.2.2. LWM2M Register Endpoint

LWM2M defines a registration interface based on the REST API, described in Section 5. The base URI path of the LWM2M Resource Directory is specified to be "/rd" as recommended in Section 5.3.

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

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

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

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

ep - Endpoint Name
lt - registration lifetime

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

Additional optional LWM2M registration parameters are defined:

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

Table 5: LWM2M Additional Registration Parameters

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

et - Endpoint Type
con - Context

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

Here is an example LWM2M registration payload:

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

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

10.2.3. LWM2M Update Endpoint Registration

An LWM2M Registration update proceeds as described in Section 5.4.1, and adds some optional parameter updates:

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

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

10.2.4. LWM2M De-Register Endpoint

LWM2M allows for de-registration using the delete method on the returned location from the initial registration operation. LWM2M de-registration proceeds as described in Section 5.4.2.

11. Acknowledgments

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

12. Changelog

changes from -09 to -10

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

o removed all text concerning DNS-SD.

o removed inconsistency in RDAO text.

o suggestions taken over from various sources

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

o moved simple registration to registration section

changes from -08 to -09

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

o changed "ins" ABNF notation.

o various editorial improvements, including in examples

o clarifications for RDAO

changes from -07 to -08

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

- 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

o Clarified endpoint name usage

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

Changes from -02 to -03:

o Added an example for lighting and DNS integration

o Added an example for RD use in OMA LWM2M

o Added Read Links operation for link inspection by endpoints

o Expanded DNS-SD section

o Added draft authors Peter van der Stok and Michael Koster

Changes from -01 to -02:

o Added a catalogue use case.

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

o Additional examples section added for more complex use cases.

o New DNS-SD mapping section.

o Added text on endpoint identification and authentication.

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

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

Changes from -00 to -01:

o Removed the ETag validation feature.

o Place holder for the DNS-SD mapping section.

o Explicitly disabled GET or POST on returned Location.

o New registry for RD parameters.

o Added support for the JSON Link Format.

o Added reference to the Groupcomm WG draft.

Changes from -05 to WG Document -00:

o Updated the version and date.

Changes from -04 to -05:

o Restricted Update to parameter updates.

o Added pagination support for the Lookup interface.

o Minor editing, bug fixes and reference updates.

o Added group support.

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

Changes from -03 to -04:

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

o Integrated the Simple Directory Discovery from Carsten.

o Editorial improvements.

o Fixed the use of ETags.

o Fixed tickets 383 and 372

Changes from -02 to -03:

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

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

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

o Improved the security considerations section.

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

Changes from -01 to -02:

o Added a terminology section.

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

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

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

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

13. References

13.1. Normative References

[I-D.ietf-core-links-json]
Li, K., Rahman, A., and C. Bormann, "Representing CoRE Formats in JSON and CBOR", draft-ietf-core-links-json-06 (work in progress), July 2016.

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

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

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

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

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

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

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

13.2. Informative References

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

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

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

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

Authors' Addresses

Zach Shelby
ARM
150 Rose Orchard
San Jose 95134
USA

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

Michael Koster
SmartThings
665 Clyde Avenue
Mountain View 94043
USA

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

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

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

Peter van der Stok
consultant

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