CoRE
Internet-Draft
Intended status: Informational
Expires: May 2, 2012

P. van der Stok
Philips Research
K. Lynn
Consultant
October 30, 2011

CoAP Utilization for Building Control
draft-vanderstok-core-bc-05

Abstract

This draft describes an example use of the RESTful CoAP protocol for building automation and control (BAC) applications such as HVAC and lighting. A few basic design assumptions are stated first, then URI structure is utilized to define group as well as unicast scope for RESTful operations.

This proposal supports the view that 1) service discovery is complementary to resource discovery and facilitates control network scaling, and 2) building control is likely to move in steps toward all-IP control networks based on the legacy efforts provided by DALI, LON, BACnet, ZigBee, and other standards.

The authority portion of the URI is used to identify a device (group) and the resulting DNS name is bound to a unicast (multicast) address. Group addressing has consequence for the naming convention of the resources of a device. Naming of URI is building or organization dependent, must be flexible, and SHOULD conform to some local convention. Naming of resources MUST be standardised preferrable by a building control related organisation.

It is shown that DNS-based service discovery can be used to locate URIs on the scale necessary in large commercial BAC deployments. The relation of DNS-SD and a Resource Directory is discussed. Finally, a method is proposed for mapping URIs onto legacy BAC resources, e.g., to discover application-layer gateways, proxies, and their dependent services.

Status of this Memo

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

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

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

This Internet-Draft will expire on May 2, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 4
  1.1. Terminology 4
  1.2. Motivation 4
2. URI structure 6
  2.1. Scheme part 7
  2.2. Authority part 7
  2.3. Path part 8
3. Group Naming and Addressing 10
4. Discovery 12
  4.1. Service discovery goals 12
  4.2. DNS-Based Service Discovery 13
  4.3. Browsing for Services 14
  4.4. Resource vs Service Discovery 14
5. DNS record structure 15
  5.1. DNS group example 18
  5.2. Operational use of DNS-SD 19
  5.3. Commissioning CoAP devices 20
    5.3.1. DNS-SD server present 21
    5.3.2. DNS-SD server not present 21
  5.4. Proxy discovery 22
6. Legacy data Representations in CoAP 23
  6.1. Network architectures 24
  6.2. Discovery of legacy gateways 26
7. Conclusions 26
8. Security considerations 27
9. IANA considerations 28
10. Acknowledgements 28
11. Changelog 28
12. References 29
  12.1. Normative References 29
  12.2. Informative References 30
Authors' Addresses 32

1. Introduction

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

In addition, the following conventions are used in this document:

A CoAP end-point, or server, is identified by a unique {IP address, port} tuple and characterised by a protocol. A server is completely specified by the authority part of a URI.

A device is the physical object that is connected to the network. A device may host one or more CoAP servers.

A service (in the service discovery sense) is a related set of resources on a CoAP server. A URI completely specifies the syntax of a service interface. Metadata describe the semantics of the service interface. The semantics may include the relation between service and the hardware connected to the device. A CoAP server may expose one or more services.

In examples below involving URIs, the authority is preceded by double slashes "//" and path is preceded by a single slash "/". The examples may make use of full or partial host names and the difference should be clear from the context.

1.2. Motivation

The CoAP protocol [I-D.ietf-core-coap] aims at providing a user application protocol architecture for a network of devices with a low resource provision such as memory, CPU capacity, and energy. In general, IT application manufacturers strive to provide the highest possible functionality and quality for a given price. In contrast, the building automation controls market is highly price sensitive and manufacturers tend to compete by delivering a given functionality and quality for the lowest price. In the first market a decreasing memory price leads to more software functionality, while in the second market it leads to a lower Bill of Material (BOM).

The vast majority of devices in a typical building control application is resource constrained, making the standardization of a lightweight application protocol like CoAP a necessary requirement for IP to penetrate the device market. The low energy consumption requirement of battery-less devices reinforces this approach. Low

resource budget implies low throughput and small packet size as for [IEEE.802.15.4]. Reduction of the packet size is obtained by using the header reduction of 6LoWPAN [RFC4944] and encouraging small payloads.

Several legacy building control standards (e.g. [BACnet], [DALI], [KNX], [LON], [ZigBee], etc.) have been developed based on years of accumulated knowledge and industry cooperation. These standards generally specify a data model, functional interfaces, packet formats, and sometimes a physical medium for data objects and function invocation. Many of these industry standards also specify proprietary transport protocols, necessitating expensive stateful gateways for these standards to interoperate. Many more recent building control network include IP-based standards for transport (at least to interconnect islands of functionality) and other functions such as naming and discovery. CoAP will be successful in the building control market to the extent that it can represent a given standard’s data objects and provide functions, e.g. resource discovery, that these standards depend on.

From the above the wanted basic syntax properties can be summarized as:

- Generate small payloads.
- Compatible with legacy standards (e.g. LON, BACnet, DALI, ZigBee Device Objects).
- Service/resource discovery in agreement with legacy standards and naming conventions.

This submission defines an approach in which the payload contains messages with a syntax defined by legacy control standards. Accordingly, the syntax of the service/resource discovery messages encapsulates legacy control standard. The intention is a progressive approach to all-IP in building control. In a first stage standard IETF based protocols (e.g. CoAP, DNS-SD) are used for transport of control messages and discovery messages expressed in a legacy syntax. This approach enables the reuse of controllers based on the semantics of the chosen control standard. In a later stage a complete redesign of the controllers can be envisaged guided by the accumulated experience with all-IP control.

Two concepts, hierarchy and group, are of prime importance in building control, particularly in lighting and HVAC. Many control messages or events are multicast from one device to a group of devices (e.g. from a light switch to all lights in an area). The scope of a multicast command or discovery message determines the group of devices that is targeted. A group scope may be defined as link-local, as a tree maintained by an IP-multicast protocol, or an

overlay that corresponds to the logical structure of a building or campus and is independent of the underlying network structure. Techniques for group communication are discussed in [I-D.rahman-core-groupcomm].

As described in "Commercial Building Applications Requirements" [I-D.martocci-6lowapp-building-applications] it is typical practice to aggregate building control at the room, area, and supervisory levels. Furthermore, networks for different subsystems (lights, HVAC, etc.) or based on different legacy standards have historically been isolated from each other in so-called "silos". RESTful web services [Fielding] represent one possible way to expose functionality and normalize data representations between silos in order to facilitate higher order applications such as campus-wide energy management.

Consequently, additional group properties are:

- Devices may be part of one or more groups.
- Resources addressed by a group must be uniformly and consistently named across all targeted devices.

For clarity, this I-D limits itself to two types of applications: (1) M2M control applications running within a building area without any human intervention after commissioning of a given network segment and (2) maintenance oriented applications where data are collected from devices in several building areas by devices inside or outside the building, and humans may intervene to change control settings. This I-D compares commercial building solutions with solutions for the home.

2. URI structure

This I-D considers three elements of the URI: scheme, authority, and path, as defined in "Uniform Resource Identifier (URI): Generic Syntax" [RFC3986]. The authority is defined within the context of standard DNS host naming, while the path is valid in relation to a fully qualified domain name (FQDN) plus optional port (and protocol is implicit, based on scheme). An example based on [RFC3986] is: foo://host.example.com:8042/over/there?name=ferret#nose, where "foo" is the scheme, "host.example.com:8042" is the authority, "/over/there" is the path, "name=ferret" is the query, and "nose" is the fragment. Fragments are not supported in CoAP.

2.1. Scheme part

The CoAP URI scheme syntax is specified in section 6 of [I-D.ietf-core-coap] and is compatible with the "http" scheme specification [RFC2616]. The scheme is implicit from the perspective of the service, but it indicates the protocol used to access the service to potential clients.

2.2. Authority part

The authority part is either a literal IP address or a DNS name comprised of a local part, specifying an individual device or group of devices, and a global part specifying a (sub)domain that may reflect the logical hierarchical structure of the building control network. The result is said to be a fully qualified domain name (FQDN) which is globally unique down to the group or device level. An optional port number may be included in the authority following a single colon ":" if the service port is other than the default CoAP value. The authority resolves to a {IP-address, port} tuple. The IP-address may be either unicast or multicast. The authority therefore identifies an individual server or a named group of servers.

The CoAP spec [I-D.ietf-core-coap] states "When a CoAP server is hosted by a 6LoWPAN device, it SHOULD also support a port in the 61616-61631 compressed UDP port space defined in [RFC4944]." As shown below, DNS-SD [I-D.cheshire-dnsext-dns-sd] is a viable technique for discovering dynamic host and port assignments for a given service. However, the use of dynamic ports in URIs is likely to lead to brittle (non-durable) identifiers as there is no assurance that a CoAP server will consistently acquire the same dynamic port and different {IP-address, port} tuples conventionally represent different servers.

A building can be unambiguously addressed by it GPS coordinates or more functionally by its zip or postal code. For example the Dutch Internet provider, KPN, assigns to each subscriber a host name based on its postcode. Analogously, an example authority for a building may be given by: //bldg.zipcode-localnr.Country/ or more concretely an imaginary address in the Netherlands as: //bldg.5533BA-125a.nl/. The "bldg" prefix can specify the target device within the building. Arriving at the device identified by //bldg.5533BA-125a.nl, the receiving service can parse the path portion of the URI and perform the requested actions on the specified resource.

Buildings have a logical internal structure dependent on their size and function. This ranges from a single hall without any structure to a complex building with wings, floors, offices and possibly a

structure within individual rooms. The naming of the building control equipment and the actual control strategy are intimately linked to the building structure. It is therefore natural to name the equipment based on their location within the building. Consequently, the local part of the URI identifying a piece of equipment is expressed in the building structure. An example is: //light-27.floor-1.west-wing...

This proposal assumes a minimal level of cooperation between the IT and building management infrastructure, namely the ability of the former to delegate DNS subdomains to the latter. This allows the building controls installer to implement an appropriate naming scheme with the required granularity. For institutional real estate such as a college or corporate campus, the authority might be based on the organization’s domain, e.g. //device-or-group.floor.wing.bldg.campus.example.com/. In cases where subdomain delegation is not an option, structure can still be represented in a "flat" namespace, subject to the 63 octet limit for a DNS label: //group1-floor2-west-bldg3-campus.example.com.

Most communication is device to device (M2M) within the building. Often a device needs to communicate to all devices of a given type within a given area of the building. For example a thermostat may access all radiator actuators in a zone. A light switch located at room 25b006 of floor one, expressed as: //switch0.25b006.floor1.5533BA-125a.nl/, might specify a command to light1 within the same room with //light1.25b006.floor1.5533BA-125a.nl/. This approach can lead to rather verbose URI strings in the packet, contrary to the small packet assumption. The question arises as to whether the syntax of the authority part needs to be standardized for building control. Given the naming flexibility provided by DNS, authority names for building control are more the concern of the building owner or the installer than a standardization concern.

2.3. Path part

The path identifies the addressable attributes of the service at the highest possible granularity. A set of paths defines the syntax of the service invocation and constitutes the interface description of the service. Every network service attribute is completely identified by a URI scheme://authority/path. In analogy, the path part of the URI specifies the resource of a given server. The naming of the services and their associated attributes are typically subjects for standardization. There is no widely accepted standard for uniformly naming building control services in a URI. A vigorous effort is undertaken by the oBIX working group of OASIS [oBIX], but its current impact is limited. There is also an open source point

naming effort underway called Project Haystack [HAYSTACK].

The path is constructed like a file system path name. It consists of a sequence of one or more name fields, with each field preceded with a slash, like /func1/subf2/final. The set of paths is structured as a tree. The last name in a name field sequence is called a leave of the tree, and the authority is the root of the path tree of a given host. The semantics of a given sub-tree in the path tree is specified by the Interface Description (if=) attribute described in [I-D.ietf-core-link-format]. As for file systems some tree naming with associated semantics can be standardized such as the de facto PC standard directory "documents and settings" with the sub-directories "My documents", "usradmin", etc. When a given body, e.g. XXX, has defined a name structure and semantics for the path tree, we say that "if = XXX" when the path tree conforms to the name structure defined by XXX.

When a GET method with an URI like "//t-sensor1.25b006.floor1.example.com/temperature" is sent, it represents an a priori understanding that the server with name t-sensor1 exists, provides a service of a given standard type (with associated semantics) (e.g. ZigBee temperature sensor), and that this standard type has the readable attribute: temperature. When commands are sent to a group of servers it MUST be the case that the targeted resource has the same path on all targeted servers. Therefore, it is necessary to establish at least a local uniform path naming convention to achieve this. One approach is to include the name of the standard, e.g. BACnet, as the first element in the path and then employ the standard’s chosen data scheme (in the case of BACnet, /bacnet/device/object/property).

The organization responsible for defining a given industry standard XXX (e.g. BACnet, ZigBee, etc.) can register the /.well-known/XXX prefix and specify the allowable path-names for a server of a given type. The same body also defines the "if=XXX" attribute. This allows the standards development organization responsibile for XXX to define the name space and resources associated with the prefix together with the associated semantics. The registered /.well-known/XXX URI effectively defines a standard object model, or schema, for services of the XXX application protocol. Manufacturers may optionally define proprietary resources that can be discovered dynamically using methods described below.

Although the authority part names need not always be transported, the path names MUST be transported in the CoAP packets. Therefore, path names SHOULD be as short as possible, even at the detriment of the clarity of the meaning of the path name.

3. Group Naming and Addressing

Within building control it is necessary to send the same command to a set of servers. Grouping allows to invoke the set of services with one application command to be executable within a specified time interval. Given a network configuration, the network operator needs to define an appropriate set of groups which can be mapped to the building areas. Knowledge about the hierarchical structure of the building areas may assist in defining a network architecture which encourages an efficient group communication implementation. IP-multicasting over the group is a possible approach for building control, although proxy-based methods may prove to be more appropriate in some deployments [I-D.rahman-core-groupcomm].

Example device groups become:

URI authority	Targeted group
//all.bldg6...	"all devices in building 6"
//all.west.bldg6...	"all devices in west wing, building 6"
//all.floor1.west.bldg6...	"all devices on floor 1, west wing, ..."
//all.bu036.floor1.west.bldg6...	"all devices in office bu036, ..."

The granularity of this example is for illustration rather than a recommendation. Experience will dictate the appropriate hierarchy for a given structure as well as the appropriate number of groups per subdomain. Note that in this example, the group name "all" is used to identify the group of all devices in each subdomain. In practice, "all" could name an address record in each of the DNS zones shown above and would bind to a different multicast address [RFC3596] in each zone. Highly granular multicast scopes are only practical using IPv6. The multicast address allocation strategy is beyond the scope of this I-D, but various alternatives have been proposed [RFC3306][RFC3307][RFC3956].

To illustrate the concept of multiple group names within a building, consider the definition, as done with [DALI], of scenes within the context of a floor or a single office. For example, the setting of all blue lights in office bu036 of floor 1 can be realized by multicasting a message to the group "//blue-lights.bu036.floor1". Each group is associated with a multicast IP address. Consequently, when the application specifies the sending of an "on" message to all blue lights in the office, the message is multicast to the associated IP address.

The binding of a group FQDN to a multicast address (i.e., creation of the AAAA record in the DNS zone server) happens during the

commissioning process. Resolution of the group name to a multicast address happens at restart of a device. A multicast address and associated group name in this context are assumed to be long-lived. It can happen that during operation the membership of the group changes (less or more lights) but its address is not altered and neither is its name. Group membership may be managed by a protocol such as Multicast Listener Discovery [RFC5790].

Similarly, a group can identify a set of resources of one server. For examples a device contains four I/O channels. The device hosts one server with four resources to access each of the four individual channels separately. Commonly, it is also required to access all four channels as one group. An additional path identifies the group of services. An example set of services and service-group is:

```
URI path        Targeted group
/IOchannel/1... "channel 1 of the IO channel device "
/IOchannel/2... "channel 2 of the IO channel device "
/IOchannel/3... "channel 3 of the IO channel device "
/IOchannel/4... "channel 4 of the IO channel device "
/IOchannel/...  "channel 1 to 4 of the IO channel device "
```

A group defines a set of servers possibly containing a set of resources. Grouping of the resources is provided by the device manufacturer. Grouping of the servers is supported by DNS and multicast protocols. The multicast address(es) identify the servers belonging to the group. A given server might belong to a number of groups. For example the server belonging to the "blue-lights" group in a given corridor might also belong to the groups: "whole building", "given wing", "given floor", "given corridor", and "lights in given corridor". From the perspective of a server, the main consequence of joining a group is it should accept packets for an additional IP address. The granularity of the domain names may have an impact on the complexity of the DNS infrastructure but not necessarily on the low-resource destinations or sources. Assuming that resolution of addresses only happens at device start-up, the complexity of the DNS server need not affect the responsiveness of the devices.

In summary, the authority portion of the URI resolves to an IP-address and port number, and identifies a server or group of servers. Authority naming is building or organization dependent, must be flexible, and does not require standardization efforts but SHOULD conform to some uniform convention. Path naming SHOULD conform to the naming convention of a standardization body.

4. Discovery

4.1. Service discovery goals

Service discovery in building control should rely on a minimal need for intervention by humans (or complete absence of humans) during system setup, bootstrapping, restart, configuration and daily operation. The goals for service discovery area:

Goal	Goal description
Return_instance	Return all instances of a given service type within a given domain
Group_instance	Group a set of instances within a group associated with a domain
Instance_resolution	Resolve the instance name to usable invocation information (e.g. IP address and port)
Group_resolution	resolve the group name to usable invocation information (e.g. IP address and port)

These goals are necessary to support the operation of commercial building control. Returning the instances results in a list of names. For building control these names can be any sequence of characters as long as for each service instance these names are unique within the domain. In [I-D.cheshire-dnsext-dns-sd] the office equipment in the IT domain is recommended to use understandable and human-readable names. The Home domain may have a need for human understandable names. This is not the case for the commercial building automation domain. However, uniqueness of the name is necessary for the application that needs to address the service in a consistent manner. Given the large number of devices in a building (several hundreds to thousands) scaling is an important aspect of the service discovery. A set of central DNS servers will provide the scalability. The expectation is that names need to be managed consistently by a central authority which can be supported by the DNS server. Tools will assist the installer and operator of the network to do the installation, configuration and maintenance of the network structure. Small devices will use the DNS server to learn the communication partners providing a given service within their domain and to resolve the IP addresses of the communication partners.

Within the home it is more important that the names convey the purpose of the service to the user reading the names and selecting his favored service instance. Non-unique names, although confusing, can probably be handled by the user of these names. Scalability is less of an issue because a smaller number of devices is implicated. The network in the home is probably more dynamic than its commercial counter-part, with many movements of devices and arrival or removal of devices.

Section 5 presents some examples of DNS structures to show how the choice of names influences the granularity of the discovery. In sections 5.1 and 5.3 a grouping example and a commissioning example, filling the DNS, are presented.

4.2. DNS-Based Service Discovery

DNS-Based Service Discovery (DNS-SD) defines a conventional way to configure DNS PTR, SRV, and TXT records to facilitate discovery of services within a subdomain, re-using the existing DNS infrastructure. This section gives a cursory overview of DNS-SD; see [I-D.cheshire-dnsext-dns-sd] for a complete description.

A DNS-SD service instance name is of the form <Instance>.<ServiceType>.<Location>.

The Location part of the service name is identical to the DNS subdomain part of the authority in URIs that identify the resources of this server or group and may identify a building zone as in the examples above.

The ServiceType SHOULD have the form [_subtype._sub.]_type._proto (e.g. _temp._sub._bc._udp). The _proto identifier provides a transport protocol hint as required by the SRV record definition [RFC2782] and, in the case of CoAP, it is always "_udp". The _type identifier is determined by standards development organization (SDO) and MUST be registered with dns-sd.org [dns-sd] (e.g. _bc for building control). The SDO is then free to specifiy one or more _subtype identifiers, which must be unique for a given _type (e.g. _temp). The _subtype and _type labels are separated by the literal "._sub" label.The maximum length of the type and subtype fields is 14 octets, but shorter names are encouraged to reduce packet sizes.

A PTR record with the label "_type._proto" is defined for each server in a selected domain, and this record’s value is set to the service instance name (which in turn identifies the SRV and TXT records for the CoAP server).

The Instance part of the service name may be changed during the commissioning process. It must be unique for a given ServiceType within the subdomain. The complete service name uniquely identifies an SRV and a TXT record in the DNS zone. The granularity of a service name MAY be at the group or server level, or it could represent a particular resource within a CoAP server. The SRV record contains the host (AAAA record) name and port of the service. The path part of the URI MUST be placed in the TXT record (path=) when multiple resources belong to the same service.

4.3. Browsing for Services

Devices in a given Location with given ServiceType, _type._proto, may be enumerated by sending a DNS query for PTR records named _type._proto to the authoritative server for that zone associated with the Location. A list of instance names for SRV records matching that <ServiceType>.<Location> is returned. Each SRV record contains the host name and port of a CoAP server. The IP address of the device is obtained by resolving the host name. DNS-SD also specifies an optional TXT record, having the same name as the SRV record, which can contain "key=value" attributes. Apart from defining standardized resources identified by if=XXX, the XXX organization may also define the standard "key=value" pairs present in the TXT record, e.g. type=switch. By convention, the first pair is txtver=<number> so that different versions of the XXX schema may interoperate. For example: A query is sent to DNS-SD to return all DALI lamps within the domain office5/mybuilding and with ServiceType: _lamp._sub._dali._udp. DNS-SD returns the list of all SRV records and AAAA records of the devices within the domain providing the wanted service.

4.4. Resource vs Service Discovery

Service discovery is concerned with finding the IP address, port, protocol, and possibly path of a named service. Resource discovery is a fine-grained enumeration of resources (path-names) of a server. [I-D.ietf-core-link-format] specifies a resource discovery pattern, such that sending a confirmable GET message for the /.well-known/core resource returns a set of links available from the server. These links describe resources hosted on that server.

CoAP link format can be used to enumerate attributes and populate the DNS-SD database in a semi-automated fashion. CoAP resource descriptions can be imported into DNS-SD for exposure to service discovery as described in [I-D.lynn-core-discovery-mapping]. The values stored in the DNS-SD directory are extracted from the information stored in the resource directory associated with a set of CoAP hosts [I-D.shelby-core-resource-directory]. The resources describe how the services can be manipulated in detail and in concreto.

It is assumed that a resource directory exists per 6LoWPAN [RFC4944], possibly running on the edge router. The DNS-SD provides a larger scope by storing the info of all services over a set of interconnected 6LoWPANs. Where the resource directory is possibly completely adequate for home networks, handling of multiple resource directories can be quite cumbersome for the many 6LoWPANs envisaged for offices. However, during network configuration, the resource

directory can be used as long as the DNS is not yet accessible.

The DNS-SD approach is complementary to the more fine-grained resource discovery, fits better the concept of service by discovering servers with given properties. DNS-SD supports a hierarchical approach to the naming of the services as discussed in section 3. DNS-SD provides a directory structure that scales well with the network size as shown by its present-day operation.

5. DNS record structure

An example is presented which explains the Resource Record (RR) structure on the DNS server. This section follows the mapping specified in [I-D.lynn-core-discovery-mapping], which defines how to fill the DNS-SD records from the link extension values. Suppose the services are delivered by XXX building control devices. The example subtype- and context- names are assumed to be standardized by the XXX alliance. All devices are situated in one office with location office4.bldg8.example.com. The names in the examples are more verbose than recommended to make the examples more readable. The table presents the services provided in the office control network:

service	ServiceType	Number
illumination	_OnOff_light._sub._bc._udp	4
presence	_occup_sensor._sub._bc._udp	1
temperature	_temp_sensor._sub._bc._udp	1
shading	_shade_control._sub._bc._udp	1

In DNS PTR records with as label the ServiceType have as value service instance names. The unique Instance names identify the service instances. In the example, the names contain id-x, with x in natural numbers. The names are usually created at the factory floor and somehow attached to the product. The ServiceTypes have been suffixed with .04.b8 to represent office4 in building8. The same suffix is used as PTR label to enemerate all instance of a given service, or within a given domain.

```
_OnOff_light._sub._bc._udp.04.b8   PTR id-1._OnOff_light
bc._udp.04.b8                      PTR id-1._OnOff_light
04.b8                              PTR id-1._OnOff_light
_OnOff_light._sub._bc._udp.04.b8   PTR id-2._OnOff_light
bc._udp.04.b8                      PTR id-2._OnOff_light
04.b8                              PTR id-2._OnOff_light
_OnOff_light._sub._bc._udp.04.b8   PTR id-3._OnOff_light
bc._udp.04.b8                      PTR id-3._OnOff_light
04.b8                              PTR id-3._OnOff_light
_OnOff_light._sub._bc._udp.04.b8   PTR id-4._OnOff_light
bc._udp.04.b8                      PTR id-4._OnOff_light
04.b8                              PTR id-4._OnOff_light
_occup_sensor._sub._bc._udp.04.b8  PTR id-5._occup_sensor
bc._udp.04.b8                      PTR id-5._occup_sensor
04.b8                              PTR id-5._occup_sensor
_temp_sensor._sub._bc._udp.04.b8   PTR id-6._temp_sensor
bc._udp.04.b8                      PTR id-6._temp_sensor
04.b8                              PTR id-6._temp_sensor
_shade_control._sub._bc._udp.04.b8 PTR id-7._temp_sensor
bc._udp.04.b8                      PTR id-7._temp_sensor
04.b8                              PTR id-7._temp_sensor
```

In the above example the id-x identifiers without the subtype suffix would be discriminating enough.

Discovery can be done with the following results. A query with the following argument returns

```
query argument                     result list
.04.8                              id-1._OnOff_light
                                   .......
                                   id-7._temp_sensor
_bc._udp.04.b8                     id-1._OnOff_light
                                   .......
                                   id-7._temp_sensor
_OnOff_light._sub._bc._udp.04.b8   id-1._OnOff_light
                                   .......
                                   id-4._OnOff_light
_occup_sensor._sub._bc._udp.04.b8  id-5._occup_sensor
```

When other offices are included in the database, the query argument 04.b8 selects those entries which are associated with office4 in building8 and rejects any others. The example shows clearly the query granularity that can be obtained and the care that must be exercised when defining the names of the ServiceTypes.

The service instances (value of PTR records) are the labels of the SRV, AAAA and TXT records describing the service instance. The SRV

record specifies the location (authority) and the port number. In the authority o4.b8 refers to office4 in building8. The AAAA record specifies the IP-address, while the TXT record specifies the subtype and the data representation of the legacy parser (if = ZigBee).

```
id-1._OnOff_light   SRV  light1.o4.b8.example.com Port-x
                    AAAA fdfd::1234
                    TXT  if=ZigBee
id-2._OnOff_light   SRV  light2.o4.b8.example.com Port-x
                    AAAA fdfd::1235
                    TXT  if=ZigBee
id-3._OnOff_light   SRV  light3.o4.b8.example.com Port-x
                    AAAA fdfd::1236
                    TXT  if=ZigBee
id-4._OnOff_light   SRV  light4.o4.b8.example.com Port-x
                    AAAA fdfd::1237
                    TXT  if=ZigBee
id-5._occup_sensor  SRV  occup.o4.b8.example.com  Port-x
                    AAAA fdfd::1238
                    TXT  if=ZigBee
id-6._temp_sensor   SRV  temp.o4.b8.example.com   Port-x
                    AAAA fdfd::1239
                    TXT  if=ZigBee
id-7._shade_control SRV  shade.o4.b8.example.com  Port-x
                    AAAA fdfd::1240
                    TXT  if=ZigBee
```

It is possible that the temperature sensor and occupancy sensor are delivered on one device. The consequence is that one device hosts two services. In the DNS table the four lights and the shade controller are unaffected. However, the PTR records with the occupancy and temperature sensor point to the same unique identifier id-8 that is suffixed with the name of the subtype. This example shows that the subtype suffix is needed to discriminate between the two service instances.

```
_occup_sensor._sub._bc._udp PTR id-8._occup_sensor
_temp_sensor._sub._bc._udp  PTR id-8._temp_sensor
```

Two SRV records with accompanying AAAA and TXT records describe the two servers, each providing one service, in more detail. The servers share the same IP address but are connected to different ports, and do have a different paths names. The TXT record is used to specify the path part with "path=".

```
id-8._occup_sensor SRV  occup.o4.b8.example.com Port-x
                   AAAA fdfd::1241
                   TXT  path=/os if=ZigBee
id-8._temp_sensor  SRV  temp.o4.b8.example.com  Port-y
                   AAAA fdfd::1241
                   TXT  path=/ts if=ZigBee
```

The path names /ts and /os are short names for temperature_sensor and occupancy_sensor respectively. Not all multi-function devices will use different ports for the individual functions. It is also quite common to use different IP interfaces with different IP addresses, reflected by the value of the AAAA records.

5.1. DNS group example

Another aspect is the grouping of servers. Where in the former section the names of the services are standardized names, this is less probable for the group names. Usually the group names are application specific or are standardized at the manufacturer. For example, assume that a group all-light.o4.b8.example.com is created which contains all four lights inside office4. The accompanying ServiceType can be defined as _all_light._sub._bc._udp. The ServiceType suffixed with 04.b8 points to a unique identifier defined as _all_light.04.b8, assuming that this is the only _all_light group within office 4 of building 8. The PTR record looks like:

```
_all_light._sub._bc._udp.04.b8 PTR _all_light.04.b8
```

It is assumed that the group all_light.o4.b8.example.com has received a multicast address: ff1e::148. The accompanying SRV, AAAA, and TXT RR become:

```
_all_light.04.b8 SRV  all_light.o4.b8.example.com Port-z
                 AAAA ff1e::148
                 TXT  if=ZigBee
```

When a multicast message is sent to a group, the path of the accessed resource must be strictly the same for all servers. The naming of the path is typically a responsibility for the standardisation organisations describing the command set for a given application area. However a constraint exits in the case of mult-function devices which host multiple resource of the same type. For example a device with three lamps with corresponding onoff attributes can be accessed via the three different paths:

```
/light/1/onoff
/light/2/onoff
/light/3/onoff
```

A unique path to the onoff resource of all instances of light on this device can be provided by /light/onoff. As this is logically the path to a single instance on a mono-function device. The corresponding unique paths for onoff to be used in the multicast message becomes /light/onoff. The corresponding resource records for a luminaire, named lm1, in DNS become:

```
_light._sub._bc._udp.04.b8 PTR  _all_light.04.b8
_light._sub._bc._udp.04.b8 PTR  _light_1.04.b8
_light._sub._bc._udp.04.b8 PTR  _light_2.04.b8
_light._sub._bc._udp.04.b8 PTR  _light_3.04.b8
_all_light.04.b8           SRV  all_light.o4.b8.example.com Port-x
                           AAAA ff1e::148
                           TXT  if=ZigBee path=/light
_light_1.04.b8             SRV  lm1.o4.b8.example.com       Port-z
                           AAAA fdfd::1234
                           TXT  if=ZigBee path=/light/1
_light_2.04.b8             SRV  lm1.o4.b8.example.com       Port-z
                           AAAA fdfd::1234
                           TXT  if=ZigBee path=/light/2
_light_3.04.b8             SRV  lm1.o4.b8.example.com       Port-z
                           AAAA fdfd::1234
                           TXT  if=ZigBee path=/light/3
```

The entries in DNS can be used to form groups with the light weight group management protocol and multicast listener discovery [RFC5790].

5.2. Operational use of DNS-SD

The populated DNS-SD server provides the necessary support for the applications to execute their control loops with minimum operator support. The operation of the office network can be split up in phases. In a first phase the network is commissioned, such that a relation is established between the IP address, the servicetype and the domain. The servicetype can be extracted from the link-format as described in [I-D.shelby-core-resource-directory]. After commissioning this information is stored in the DNS-SD files. In a second phase groups are formed and group names with their IP address are stored in the DNS-SD files. The IP multicast addresses are communicated to the members of the groups. In the third and final phase, applications query DNS-SD to find the IP addresses of the services within a given domain, and of the groups within a given domain.

In the home, a commissioning phase requiring the intervention of an installer (a "truck roll") is to be avoided if possible. Here the first phase consists of the booting up devices which insert their services resources to a link-format directory. The information from the resource directory can be inserted into DNS-SD or into xmDNS [I-D.lynn-dnsext-site-mdns] when appropriate. In the second phase remote controllers or other hand-held devices can be used to discover the services of a given type, to group the services, and to store the group names into DNS-SD or xmDNS as appropriate. Pointing out the members of a group can be in any kind of manner from typing members in, selecting them from a browser list, etc.

5.3. Commissioning CoAP devices

For clarity it is assumed in this section that a device hosts one server. A device has received a unique device identifier at the production plant. Given the authority naming presented in section 2.2 the authority name represents the location of the host within the building.

Commissioning means the following three actions:

- Defining the URI (location)
- Assigning an IP address to the URI
- mapping the unique device identifier to the URI

Two cases of the office network are considered for commissioning: (1) no 6LBR and no DNS server connected, and (2) a 6LBR connects the office network to a DNS server.

When an architect has designed the building and described all light points, ventilators, heating- and cooling units, and sensors, it is necessary to identify all these devices spatially and functionally. Storing the triple <Instance>.<ServiceType>.<Location> into DNS-SD represents the commissioning process. The Instance is the unique identifier given to the device in the factory but which has no relation to its later location. The ServiceType together with the Location represent the spatial and functional aspects of the device as specified by the architect.

Design decision: A commissioning tool with access to the network is used for the commissioning phase.

For example, dependent on used technology and production process, the following situation (state) may exist in a host after physical installation of the devices and before commissioning:

- A given host is unaware of its Location.

- A given host knows its ServiceType and Instance. The Instance is also readable by bar code reader.

- The commissioning tool knows all Locations to which hosts need to be assigned.

- Each host has a (site-local) IP address.

Consider the commissioning process (1) with a central DNS-SD server and (2) without a central server using xmDNS. The commissioning processes described below are just examples and should not be taken as working procedures for commissioning devices in a building.

5.3.1. DNS-SD server present

The installer reads with a bar code reader, attached to the commissioning tool, the identifier of the device to commission. It is assumed that the tool can learn the IP address of the device with the given identifier. The tool displays on a screen the physical lay-out of the devices within the building. The installer selects, on the screen of the tool, the physical location of the chosen device. From the designated physical location the tool creates the URI of the selected device. The tool inserts the URI and the IP address into the DNS server. For example the light with URI light1.o4.b8.example.com is represented with an AAAA record:

```
light1.o4.b8.example.com AAAA fdfd::1234
```

The tool reads the service name and type from the device using resource information stored according to the link-format [I-D.ietf-core-link-format]. With this information the tool constructs the PTR, SRV and TXT records according to the example presented in section 5.

This is done for all devices within a given part of the building. After the commissioning process, all resources of each device have an URI and IP address which are stored in the central DNS-SD server. When devices are restarted, the DHCP server may allocate new IP addresses to the device and update the DNS server.

5.3.2. DNS-SD server not present

It is assumed that the building network is composed of independent network segments (possibly a single site) such that each device on a given segment can communicate directly with any other device on this segment. The segments are not connected to a 6LBR and have no access

to DNS or other servers. The installer knows these segments and has a list of devices for a given segment. In the tool the installer selects the names which belong to the given building segment. The selected names are converted to site-local authorities and stored in the tool. All devices are assumed to have selected a site-local IP address. Assume that every device has a unique barcode within the building and that the corresponding device knows the bar code number. The installer reads with a bar code reader, attached to the tool, the Instance name of the device to commission. The installer selects, on the screen of the tool, the physical location of the chosen device. The tool knows the authority of the selected device. The tool broadcasts the bar code number and authority to all connected devices. The device with the given barcode number, extends the authority with the path name of the resources. For each resource, the device multicasts the site-local IP-address and the site-local URI to the xmDNS servers in the connected devices. This concludes the commissioning of a network segment. All resources of each device have a site-local URI and a site-local IP address which are stored in the xmDNS servers.

5.4. Proxy discovery

Proxies will be used in CoAP networks for at least two major reasons: (1) http/coap proxy, and (2) proxy of service on battery-less device. The first proxy is probably implemented as forward proxy, while the latter is probably implemented as backward proxy. The battery-less device will at rare occasions (when it is not sleeping) and during installation answer the GET /.well-known/core request. The return data are used by the installation tool to make the proxy device return the same resource names on /.well-known/core as is returned by the sleeping device. An installation tool installs on the proxy all the resources of the sleeping device for which the proxy is assumed to answer. Consequently, the proxy is discovered as a multi-server host with as many path names as it proxies sleeping servers. The servers on sleeping devices should not be discoverable via DNS-SD. However, AAAA records are generated for the sleeping device host name. This host name is used by the proxy to subscribe to the "sporadic" services of the sleeping device. For example assume two sleeping devices, an occupancy sensor and a temperature sensor, and one proxy. Two service types are defined with PTR records in DNS-SD. The identifier id-1 of the proxy is used by the installation tool to define the Instances.

```
_occup_sensor._sub._bc._udp.04.b8 PTR id-1._occup_sensor
_temp_sensor._sub._bc._udp.04.b8  PTR id-1._temp_sensor
```

Two SRV records with accompanying AAAA and TXT records describe the two services in more detail. The services share the same IP address,

are connected to the same port, but do have different paths names. The TXT record is used to specify the path part with "path=".

```
id-1._occup_sensor      SRV  proxy.o4.b8.example.com Port-x
                        AAAA fdfd:: 1241
                        TXT  path=/os if=ZigBee
id-1._temp_sensor       SRV  proxy.o4.b8.example.com Port-x
                        AAAA fdfd:: 1241
                        TXT  path=/ts if=ZigBee
sl-ts.o4.b8.example.com AAAA fdfd::1242
sl-os.o4.b8.example.com AAAA fdfd::1243
```

The path names /ts and /os are short names for temperature_sensor and occupancy_sensor respectively and were taken over from link-format information contained in the sleeping devices. Two AAAA records are provided for the two sleeping devices. The proxy has used the domain names of the sleeping devices to subscribe to the publications of the two sleeping devices.

It is important to remark that there are now two services with the same resources present on two different devices: the sleeping device and its proxy. When a host invokes the /.well-known/core resource, it should be possible to distinguish between the proxy (to be invoked) and the sleeping device (not to be invoked). The distinction is necessary once the sleeping device is discoverable and the sleeping device is awake from time to time. It is suggested that the link-format syntax allows to make this distinction.

6. Legacy data Representations in CoAP

Before CoAP devices can come to market, manufacturers must agree that the type and resources of the device can be interpreted according to some generally recognized syntax. At this moment no such generally recognized syntax exists for CoAP devices. We do not expect an IETF working group to standardize such a syntax, and we are convinced that syntax standardization is the responsibility of industry standards organizations. Given the long history of building control, many groups have defined a data representation for building control devices for example BACnet, ZigBee, oBIX, LON, KNX, and many others. It is our belief that new representations will be defined and must coexist with the named legacy ones.

The CoAP protocol should transport any data representation, and certainly the legacy ones.It is expected that a CoAP client can handle one or more legacy representation. Given that a CoAP client can handle representation of standard XXX, this I-D proposes that such a CoAP device can communicate with legacy devices via a CoAP/

legacy gateway (router).

6.1. Network architectures

```
+------+                +------+     +------+
| yyy  |                | zzz  |     | zzz  |
| link |                | CoAP |     | CoAP |
+------+                +------+     +------+
  |         +---------+
  |---------| CoAP    |-->
  |         | gateway |
+------+    +---------+        +-----------+
| yyy  |                       | zzz ; yyy |
| link |                       |   CoAP    |
+------+                       +-----------+
```

Figure 1: network with multiple representation standards

Figure 1 represents the network architecture which is expected for the purpose of this I-D. The CoAP gateway connects one link with two legacy devices -containing legacy data representation "yyy"- with the wireless CoAP network composed of three CoAP hosts. Two CoAP hosts contain the CoAP stack with a zzz representation and one host contains the CoAP stack with a zzz and an yyy representation. The yyy hosts can freely communicate according to the yyy link protocol over the yyy link. The zzz CoAP hosts, including the zzz;yyy host can freely exchange zzz data representations according to the CoAP protocol over the wireless 6LoWPAN network. The zzz;yyy host can send yyy data representations to the CoAP gateway which passes them on to the specified yyy legacy host. The yyy legacy device returns data to the requesting zzz;yyy CoAP host via the same gateway.

The CoAP hosts can address the legacy devices behind the gateway in at least 4 ways.

- All devices of legacy network YYY share the URI with the CoAP gateway. Every legacy device is a resource for the gateway as seen from the CoAP host. Consequently, the CoAP host sends the message to the IP address of the gateway and the gateway parses the URI-Path to determine the specified legacy device.

- All devices of legacy network YYY have IP addresses different from the IP address of the gateway. Consequently, a CoAP host sends the message to the IP address of the specified device. The routing protocol on the CoAP network makes the message arrive at the CoAP gateway. The gateway determines the specified legacy device from the destination IP address.

- All devices of legacy network YYY have different authorities. The authorities of the legacy device resolve to an IP address of the gateway. This means that the possibly lengthy authority names need to be transmitted. The gateway recognizes the authorities and maps authority to legacy device.

- All devices of legacy network YYY have different ports. This can be expressed in two ways (1) as :port in the URI, or (2) in the DNS-SD records. In the latter case the port is defined in the UDP header and is efficient in packet header size.

The major advantage of all four approaches is that the gateway only handles the URI or IP address and port number to select the destination legacy device independent of the type of legacy device and the contents of the legacy payload of the message. In Figure 1 the gateway connects to a single link. For example, this would be the case for DALI standard. Other legacy standards, like BACnet, LON, allow networks composed of multiple links.

An example of an invocation of a ZZZ service (See figure 2). The resource path /ZZZ identifies the parser of the ZZZ syntax. A 12 octet string completely describes the ZZZ command. The host is completely identified by the authority in the URI. The ZZZ parser on the host is identified by the port number in the UDP header (not shown).

```
    Client                                            CoAP/ZZZ
      |                                                 device
      |  REQUEST                                          |
      |-------- CON [0x5577] PUT /ZZZ -------->|
      |             binary 12 octet string                |
      |                                                   |
      |  RESPONSE                                         |
      |<---------- ACK [0x5577] 2.00 OK  ----------------- |
      |                                                   |
```

Figure 2: Sending a ZZZ command with CoAP to CoAP/ZZZ device

An example of an invocation of a DALI legacy device behind a gateway is given in figure 3. The resource path /DALI identifies the DALI parser. The application sets a value of 200 in the DALI device in the resource 256 defined by the DALI spec.

```
       Client                                            DALI/CoAP
         |                                                gateway
         |  REQUEST                                          |
         |------- CON [0x5577] PUT /DALI -------->|
         | binary 16 bit payload dt*256 + 200                |
         |                                                   |
         |  RESPONSE                                         |
         |<---------- ACK [0x5577] 2.00 OK  ---------------- |
         |                                                   |
```

Figure 3: Sending a DALI setting with CoAP to CoAP/DALI gateway

6.2. Discovery of legacy gateways

Discovery of legacy gateways is not very different from discovery of proxies in section 5.4. the consequences for discovery are listed for the four modes of addressing legacy devices via a gateway of section 6.1.

- The gateway presents a list of resources representing the legacy devices. Discovery is done as for other CoAP devices.

- Each legacy device has a different IP address. The gateway must create entries in the DNS for as many legacy devices. The authority of the legacy device is the authority of the gateway with a ServiceType to be specified by the gateway.

- All devices of legacy network YYY have different authorities. In this case each legacy device has the same IP address as the gateway. The gateway must create entries in the DNS for as many legacy devices.

- All devices of legacy network YYY have different ports. The gateway must create entries in the DNS for as many legacy devices. Each entry has the authority of the gateway with a different ServiceType and a different port number.

7. Conclusions

This I-D explains how naming in building control is based on a hierarchical structure of the building areas. It is shown that DNS naming can be used to express this hierarchy in the authority portion of the URI, down to the group or device level. The hierarchical naming scheme need not be standardized, but rather can be designed to suit the application. However, it is recommended that the scheme be employed consistently throughout the delegated subdomain(s).

The authority portion of the URI is resolved by the client, using conventional DNS, into the unicast or multicast IP address of the targeted device(s). Taking advantage of the CoAP design [I-D.ietf-core-coap], the URI-Host option need not be transmitted in requests to origin servers and thus there is no performance penalty for using descriptive naming schemes. The CoAP design allows sending a short URI to distinguish between resources on a given device, resulting in very compact identifiers.

DNS-SD [I-D.cheshire-dnsext-dns-sd] can be used to scale up service discovery beyond the 6LoWPAN. DNS-SD can be used to enumerate instances of a given service type within a given sub-domain. This affords additional flexibility, such as the ability to discover dynamic port assignments for CoAP device, locate CoAP devices by subtype, or bind service names for particular CoAP URIs.

This I-D discusses the addressing, discovery and naming of legacy devices behind gateways. The discovery of backward proxies of sleeping devices is handled in a similar fashion.

A targeted resource is specified by the path portion of the URI. Again, this I-D does not mandate a universal naming standard for resources but uses examples to show how resources could be named for various legacy standards. An obvious requirement for resources that are accessed by multicast is that they MUST all share the same path. It is shown that it is possible to transport legacy commands (e.g. expressed in BACnet, LON, DALI, ZigBee, etc.) inside a CoAP message body. Entering ServiceTypes particular to a given standard necessitates that the standardization body declares the ServiceType to dns.org.

8. Security considerations

TBD: The detailed CoAP security analysis needs to encompass scenarios for building control applications.

Based on the programming model presented in this I-D, security scenarios for building control need to be stated. Appropriate methods to counteract the proposed threats may be based on the work done elsewhere, for example in the ZigBee over IP context.

Multicast messages are, by their nature, transmitted via UDP. Any privacy applied to such messages must be block oriented and based on group keys shared by all targeted devices. The CoRE security analysis must be broadened to include multicast scenarios.

9. IANA considerations

This I-D proposes that associations which standardize device representations (like BACnet, ZigBee, DALI,...) contact IANA to reserve the prefix /.well-known/XXX for the standard XXX.

10. Acknowledgements

This I-D has benefited from conversations with and comments from Andrew Tokmakoff, Emmanuel Frimout, Jamie Mc Cormack, Oscar Garcia, Dee Denteneer, Joop Talstra, Zach Shelby, Jerald Martocci, Anders Brandt, Matthieu Vial, Jerome Hamel, George Yianni, and Nicolas Riou.

11. Changelog

From bc-01 to bc-02

- Removed all references to multicast and multicast scope, given draft of rahman group communication.

- Adapted examples to CoAP-2 and core-link drafts.

- transport short URL for destination recognition.

- Elaborated legacy discovery under DNS-SD.

From bc-02 to bc-03

- Elaboration on gateways, commissioning and legacy networks.

- Recommendation to extend DNS-SD naming with sn, st, and ss attributes.

From bc-03 to bc-04

- moved core link extension sub-section to discovery mapping draft

- extended use of service type

- gave DNS record examples and worked out multifunction device

- added proxy discovery and legacy gateway discovery

- defined path tree and corresponding schema

- reviewed definition of group, device, server, service (interface),

resopurce, and attribute.

From bc-04 to bc-05

- extended and corrected examples for multi-function devicesw

- syntax more compatible with other resource discovery I-Ds

- abstract adapted

- more stringent use of the words server, end point, service and devices

12. References

12.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.

[RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, August 2002.

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, November 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, March 2008.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010.

12.2. Informative References

[I-D.cheshire-dnsext-dns-sd]
Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", draft-cheshire-dnsext-dns-sd-10 (work in progress), February 2011.

[I-D.cheshire-dnsext-multicastdns]
Cheshire, S. and M. Krochmal, "Multicast DNS", draft-cheshire-dnsext-multicastdns-14 (work in progress), February 2011.

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-07 (work in progress), July 2011.

[I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format", draft-ietf-core-link-format-07 (work in progress), July 2011.

[I-D.martocci-6lowapp-building-applications]
Martocci, J., Schoofs, A., and P. Stok, "Commercial

Building Applications Requirements", draft-martocci-6lowapp-building-applications-01 (work in progress), July 2010.

[I-D.rahman-core-groupcomm]
Rahman, A. and E. Dijk, "Group Communication for CoAP", draft-rahman-core-groupcomm-07 (work in progress), October 2011.

[I-D.shelby-core-resource-directory]
Shelby, Z. and S. Krco, "CoRE Resource Directory", draft-shelby-core-resource-directory-01 (work in progress), September 2011.

[I-D.lynn-core-discovery-mapping]
Lynn, K. and Z. Shelby, "CoRE Link-Format to DNS-Based Service Discovery Mapping", draft-lynn-core-discovery-mapping-01 (work in progress), July 2011.

[I-D.lynn-dnsext-site-mdns]
Lynn, K. and D. Sturek, "Extended Multicast DNS", draft-lynn-dnsext-site-mdns-01 (work in progress), March 2011.

[BACnet] Bender, J. and M. Newman, "BACnet/IP", Web http://www.bacnet.org/Tutorial/BACnetIP/index.html, 2000.

[ZigBee] Tolle, G., "A UDP/IP Adaptation of the ZigBee Application Protocol", draft-tolle-cap-00 (work in progress), October 2008.

[LON] "LONTalk protocol specification, version 3", 1994.

[DALI] "DALI Manual", Web http://www.dali-ag.org/c/manual_gb.pdf, 2001.

[KNX] Kastner, W., Neugschwandtner, G., and M. Koegler, "AN OPEN APPROACH TO EIB/KNX SOFTWARE DEVELOPMENT", Web http://www.auto.tuwien.ac.at/~gneugsch/fet05-openapproach-preprint.pdf, 2005.

[IEEE.802.15.4]
"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical

Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)", IEEE Std 802.15.4-2006, June 2006, <http://standards.ieee.org/getieee802/802.15.html>.

[HAYSTACK] "Project Haystack", Web http://project-haystack.org/, 2011.

[oBIX] Frank, B., Ed., "oBIX working group", Web http://www.obix.org, 2006.

[Fielding] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures, Second Edition", Doctoral dissertation , University of California, Irvine , Web http://www.ics.uci.edu/~fielding/pubs/dissertation/top.html, 2000.

[ZigBee-IP] "ZigBee Smart Energy Profile 2.0 Application Protocol Specification", Draft ZigBee-11167, March 2011.

[dns-sd] "dns-sd servicetype registration", Web http://www.dns-sd.org/ServiceTypes.html, 2011.

Authors' Addresses

Peter van der Stok
Philips Research
High Tech Campus 34-1
Eindhoven, 5656 AA
The Netherlands

Email: peter.van.der.stok@philips.com

Kerry Lynn
Consultant

Phone: +1 978 460 4253
Email: kerlyn@ieee.org