

This Internet-Draft, draft-bormann-core-coap-block-01.txt, has been replaced by another document, draft-ietf-core-block-00.txt, and has been deleted from the Internet-Drafts directory.

Internet-Drafts are not archival documents, and copies of Internet-Drafts that have been deleted from the directory are not available. The Secretariat does not have any information regarding the future plans of the author(s) or working group, if applicable, with respect to this deleted Internet-Draft. For more information, or to request a copy of the document, please contact the author(s) directly.

Draft Author(s):

C. Bormann <cabo@tzi.org>

CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2011

Z. Shelby, Ed.
Sensinode
C. Bormann
Universitaet Bremen TZI
October 18, 2010

Blockwise transfers in CoAP
draft-ietf-core-block-00

Abstract

CoAP is a RESTful transfer protocol for constrained nodes and networks. CoAP is based on datagram transport, which limits the maximum size of resource representations that can be transferred without too much fragmentation. The Block option provides a minimal way to transfer larger representations in a block-wise fashion.

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 April 21, 2011.

Copyright Notice

Copyright (c) 2010 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	3
2. Block-wise transfers	4
2.1. The Block Option	4
2.2. Using the Block Option	6
3. IANA Considerations	9
4. Security Considerations	10
4.1. Mitigating Amplification Attacks	10
5. Acknowledgements	11
6. References	12
6.1. Normative References	12
6.2. Informative References	12
Authors' Addresses	13

1. Introduction

The CoRE WG is tasked with standardizing an Application Protocol for Constrained Networks/Nodes, CoAP. This protocol is intended to provide RESTful [REST] services not unlike HTTP [RFC2616], while reducing the complexity of implementation as well as the size of packets exchanged in order to make these services useful in a highly constrained network of themselves highly constrained nodes.

This objective requires restraint in a number of sometimes conflicting ways:

- o reducing implementation complexity in order to minimize code size,
- o reducing message sizes in order to minimize the number of fragments needed for each message (in turn to maximize the probability of delivery of the message), the amount of transmission power needed and the loading of the limited-bandwidth channel,
- o reducing requirements on the environment such as stable storage, good sources of randomness or user interaction capabilities.

CoAP is based on datagram transports such as UDP, which limit the maximum size of resource representations that can be transferred without creating unreasonable levels of fragmentation. The Block option provides a minimal way to transfer larger resource representations in a block-wise fashion.

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP implementations.

2. Block-wise transfers

Not all resource representations will fit into a single link layer packet of a constrained network. Using fragmentation (either at the adaptation layer or at the IP layer) to enable the transport of larger representations is possible up to the maximum size of the underlying datagram protocol (such as UDP), but the fragmentation/reassembly process loads the lower layers with conversation state that is better managed in the application layer.

This specification proposes an option to enable `_block-wise_` access to resource representations. The overriding objective is to avoid creating conversation state at the server for block-wise GET requests. (It is impossible to fully avoid creating conversation state for POST/PUT, if the creation/replacement of resources is to be atomic; where that property is not needed, there is no need to create server conversation state in this case, either.)

Implementation of the Block option is intended to be optional. However, when it is present in a CoAP message, it **MUST** be processed; therefore it is identified as a critical option.

The size of the blocks should not be fixed by the protocol. On the other hand, implementation should be as simple as possible. The Block option therefore supports a small range of power-of-two block sizes, from 2^4 (16) to 2^{11} (2048) bytes. One of these eight values can be encoded in three bits (0 for 2^4 to 7 for 2^{11} bytes), the "SZX" (size exponent); the actual block size is then " $1 \ll (\text{SZX} + 4)$ ".

2.1. The Block Option

When a representation is larger than can be comfortably transferred in a single UDP datagram, the Block option can be used to indicate a block-wise transfer. Block is a 1-, 2- or 3-byte integer, the four least significant bits of which indicate the size and whether the current block-wise transfer is the last block being transferred (M or "more" bit). The value divided by sixteen is the number of the block currently being transferred, starting from zero, i.e., the current transfer is about the "size" bytes starting at "block number $\ll (\text{SZX} + 4)$ ". The default value of the Block Option is zero, indicating that the current block is the first (block number 0) and only (M bit not set) block of the transfer; however, there is no explicit size implied by this default value.

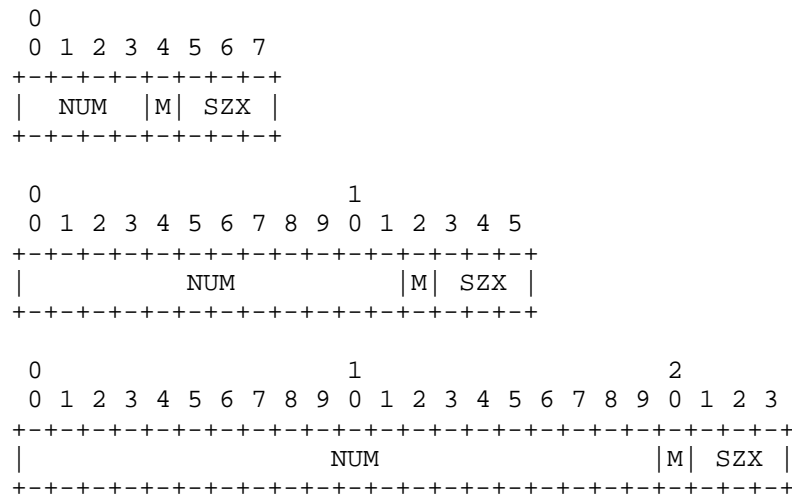


Figure 1: Block option

(Note that the option with the last 4 bits masked out, shifted to the left by the value of SZX, gives the byte position of the block.)

NUM: Block Number. The block number is a variable 4-20 bit unsigned integer indicating the block number being requested or provided. Block number 0 indicates the first block of a representation.

M: More Flag. This flag indicates if this block is the last in a representation when set. When not set it indicates that there are one or more blocks available. When the block option is used to retrieve a specific block number the M bit **MUST** be sent as zero and ignored on reception.

SZX: Block Size. The block size indicates the size of a block to the power of two. Thus block size = $2^{(SZX + 4)}$. Thus the minimum block size is 16 and the maximum is 2048.

The Block option is used in one of three roles:

- o In the request for a GET, the Block option gives the block number requested and suggests a block size (block number 0) or echoes the block size of previous blocks received (block numbers other than 0).
- o In the response for a GET or in the request for a PUT or POST, the Block option describes what block number is contained in the payload, and whether further blocks are part of that body (M bit). If the M bit is set, the size of the payload body in bytes **MUST**

indeed be the power of two given by the block size. All blocks for a REST transaction MUST use the same block size, except for the last block (M bit not set).

- o In the response for a PUT or POST, the Block option indicates what block number is being acknowledged. In this case, the M bit is set to indicate that this response does not carry the final response to the request; this can occur when the M bit was set in the request and the server implements PUT/POST atomically (i.e., acts only upon reception of the last block).

2.2. Using the Block Option

Using the Block option, a single REST operation can be split into multiple CoAP message transactions. Each of these message transactions uses their own CoAP transaction ID.

When a GET is answered with a response carrying a Block option with the M bit set, the requestor may retrieve additional blocks of the resource representation by sending requests with a Block option giving the block number desired. In such a Block option, the M bit MUST be sent as zero and ignored on reception.

To influence the block size used in response to a GET request, the requestor uses the Block option, giving the desired size, a block number of zero and an M bit of zero. A server SHOULD use the block size indicated or a smaller size. Any further block-wise requests for blocks beyond the first one MUST indicate the same block size that was already used in the response for the first one.

If the Block option is used by the requestor, all GET requests in a single transaction (except for the last one with the M bit not set) MUST ultimately use the same size. The server SHOULD use the block size indicated in the request option or a smaller size, but the requestor MUST take note of the actual block size used in the response it receives to its initial GET and proceed to use it in subsequent GETs; the server behavior MUST ensure that this client behavior results in the same block size for all responses in a sequence (except for the last one with the M bit not set).

Block-wise transfers can be used to GET resources the representations of which are entirely static (not changing over time at all, such as in a schema describing a device), or for dynamically changing resources. In the latter case, the Block option SHOULD be used in conjunction with the Etag option, to ensure that the blocks being reassembled are from the same version of the representation. When reassembling the representation from the blocks being exchanged, the reassembler MUST compare Etag options. If the Etag options do not

match in a GET transfer, the requestor has the option of attempting to retrieve fresh values for the blocks it retrieved first. To minimize the resulting inefficiency, the server MAY cache the current value of a representation for an ongoing sequence of requests, but there is no requirement for the server to establish any state. The client MAY facilitate identifying the sequence by using the Token option.

In a PUT or POST transfer, the Block option refers to the body in the request, i.e., there is no way to perform a block-wise retrieval of the body of the response. Servers that do need to supply large bodies in response to PUT/POST SHOULD therefore be employing redirects.

In a PUT or POST transfer that is intended to be implemented in an atomic fashion at the server, the actual creation/replacement takes place at the time a block with the M bit unset is received. If not all previous blocks are available at the server at this time, the transfer fails and error code 4__ (TBD) MUST be returned. The error code 4__ can also be returned at any time by a server that does not currently have the resources to store blocks for a block-wise PUT or POST transfer that it would intend to implement in an atomic fashion.

If multiple concurrently proceeding block-wise PUT or POST operations are possible, the requestor SHOULD use the Token option to clearly separate the different sequences. In this case, when reassembling the representation from the blocks being exchanged to enable atomic processing, the reassembler MUST compare any Token options present (taking an absent Token option to default to the empty Token). If atomic processing is not desired, there is no need to check the Token option.

In summary, this specification: Adds a Block Option that can be used for block-wise transfers.

Benefits: Transfers larger than can be accommodated in constrained-network link-layer packets can be performed in smaller blocks.

No hard-to-manage conversation state is created at the adaptation layer or IP layer for fragmentation.

The transfer of each block is acknowledged, enabling retransmission if required.

Both sides have a say in the block size that actually will be used.

TBD: Give examples with detailed message flows for a block-wise GET,
PUT and POST.

3. IANA Considerations

This draft adds the following option number to Table 2 of [I-D.ietf-core-coap]:

Type	C/E	Name	Data type	Length	Default
13	C	Block	Unsigned Integer	1-3 B	0 (see Section 2.1)

4. Security Considerations

TBD. (Weigh the security implications of application layer block-wise transfer against those of adaptation-layer or IP-layer fragmentation.)

4.1. Mitigating Amplification Attacks

TBD. (This section discusses how CoAP nodes could become implicated in DoS attacks by using the amplifying properties of the protocol, as well as mitigations for this threat.)

A CoAP server can reduce the amount of amplification it provides to an attacker by offering large resource representations only in relatively small blocks. E.g., for a 1000 byte resource, a 10-byte request might result in an 80-byte response (with a 64-byte block) instead of a 1016-byte response, considerably reducing the amplification provided.

5. Acknowledgements

Of course, much of the content of this draft is the result of discussions with the [I-D.ietf-core-coap] authors. Tokens were suggested by Gilman Tolle and refined by Klaus Hartke.

6. References

6.1. Normative References

- [I-D.ietf-core-coap] Shelby, Z., Frank, B., and D. Sturek, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-02 (work in progress), September 2010.
- [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.

6.2. Informative References

- [REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000.

Authors' Addresses

Zach Shelby (editor)
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

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

Phone: +49-421-218-63921
Fax: +49-421-218-7000
Email: cabo@tzi.org

CoRE
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2011

Z. Shelby
Sensinode
B. Frank
SkyFoundry
D. Sturek
Pacific Gas & Electric
October 26, 2010

Constrained Application Protocol (CoAP)
draft-ietf-core-coap-03

Abstract

This document specifies the Constrained Application Protocol (CoAP), a specialized web transfer protocol for use with constrained networks and nodes for machine-to-machine applications such as smart energy and building automation. These constrained nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of kbit/s. CoAP provides a method/response interaction model between application end-points, supports built-in resource discovery, and includes key web concepts such as URIs and content-types. CoAP easily translates to HTTP for integration with the web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments.

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 April 29, 2011.

Copyright Notice

Copyright (c) 2010 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
2. Constrained Application Protocol	5
2.1. Interaction Model	5
2.1.1. Synchronous response	6
2.1.2. Asynchronous response	6
2.2. Transaction messages	8
2.2.1. Confirmable (CON)	8
2.2.2. Non-Confirmable (NON)	8
2.2.3. Acknowledgment (ACK)	8
2.2.4. Reset (RST)	8
2.2.5. Transaction IDs	9
2.3. Methods	9
2.3.1. GET	9
2.3.2. POST	9
2.3.3. PUT	10
2.3.4. DELETE	10
2.4. Response Codes	10
2.5. Options	10
2.5.1. Option Processing	10
2.5.2. URIs	11
2.5.3. Content-type encoding	12
3. Message Formats	12
3.1. CoAP header	13
3.2. Header options	14
3.2.1. Content-type Option	16
3.2.2. Uri-Authority Option	16
3.2.3. Uri-Path Option	16
3.2.4. Uri-Query Option	17
3.2.5. Location Option	17
3.2.6. Max-age Option	17
3.2.7. Etag Option	17
3.2.8. Token Option	17
4. UDP Binding	18

4.1.	Multicast	19
4.2.	Retransmission	19
4.3.	Congestion Control	19
4.4.	Default Port	19
5.	Caching	20
5.1.	Cache control	20
5.2.	Cache refresh	21
5.3.	Proxying	21
6.	Resource Discovery	22
7.	HTTP Mapping	22
8.	Protocol Constants	23
9.	Examples	23
10.	Security Considerations	26
10.1.	Securing CoAP with IPsec	27
10.2.	Securing CoAP with DTLS	28
10.2.1.	SharedKey & MultiKey Modes	28
10.2.2.	Certificate Mode	28
10.3.	Threat analysis and protocol limitations	29
10.3.1.	Processing URIs	29
10.3.2.	Proxying and Caching	29
10.3.3.	Risk of amplification	30
11.	IANA Considerations	31
11.1.	Codes	31
11.2.	Content Types	32
12.	Acknowledgments	33
13.	Changelog	34
14.	References	36
14.1.	Normative References	36
14.2.	Informative References	37
	Authors' Addresses	38

1. Introduction

The use of web services on the Internet has become ubiquitous in most applications, and depends on the fundamental Representational State Transfer (REST) architecture of the web.

The Constrained RESTful Environments (CoRE) working group 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). Constrained networks like 6LoWPAN support the expensive fragmentation of IPv6 packets into small link-layer frames. One design goal of CoRE has been to keep message overhead small, thus limiting the use of fragmentation.

One of the main goals of CoRE is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation and other M2M applications. The goal of CoAP is not to blindly compress HTTP, but rather to realize a subset of REST common with HTTP but optimized for M2M applications. Although CoRE could be used for compressing simple HTTP interfaces, it more importantly also offers features for M2M such as built-in discovery, multicast support and asynchronous transactions.

This document specifies the Constrained Application Protocol (CoAP) , which easily translates to HTTP for integration with the existing web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments and M2M applications [I-D.shelby-core-coap-req]. CoAP has the following main features:

- o Constrained web protocol fulfilling M2M requirements.
- o A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP.
- o UDP binding with reliable unicast and best-effort multicast support.
- o Asynchronous transaction support.
- o Low header overhead and parsing complexity.
- o URI and Content-type support.
- o Built-in resource discovery.

- o Simple proxy and caching capabilities.

2. Constrained Application Protocol

This section specifies the basic functionality and processing rules of the protocol.

2.1. Interaction Model

The interaction model of CoAP is similar to the client/server model of HTTP. However, Machine-to-machine interactions typically result in a CoAP implementation acting in both client and server roles (called an end-point). A CoAP exchange is equivalent to that of HTTP, and is sent by a client to request an action (using a Method Code) on a resource (identified by a URI) on a server. A response is then sent with a Response Code and resource representation if appropriate.

Unlike HTTP, CoAP deals with these interchanges asynchronously over a UDP transport with support for both unicast and multicast interactions. This is achieved using transaction messages (Confirmable, Non-Confirmable, Acknowledgment, Reset) supporting optional reliability (with exponential back-off) and transaction IDs between end-points to carry requests and responses. These transactions are transparent to the request/response interchanges. The only difference being that responses may arrive asynchronously.

One could think of CoAP as using a two-layer approach, a transactional layer used to deal with UDP and the asynchronous nature of the interactions, and the request/response interactions using Method and Response codes.

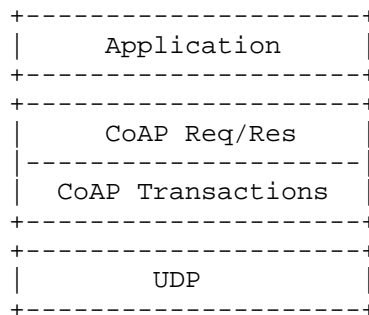


Figure 1: Abstract layering of CoAP

2.1.1.1. Synchronous response

The most basic interaction between the Req/Res and Transaction layers works by sending a request in a confirmable CoAP message and waiting for an acknowledgment message that also carries the response. E.g., two possible interactions for a basic GET are shown in Figure 2.

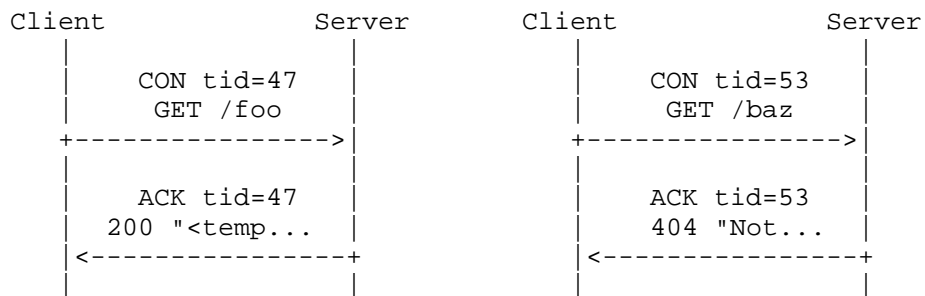


Figure 2: Two basic GET transactions, one successful, one not found

Note that at the transaction layer, the response is returned in an ACK message, independent of whether the request was successful at the Req/Res layer. In effect, the response is piggy-backed on the ACK message, so no separate acknowledgment is required that the GET message was received.

The relationship between the confirmable message (CON) and the acknowledgment message (ACK) is indicated by the transaction ID, which is echoed back by the server in the ACK. Transaction IDs are short-lived, they only serve to couple CON and ACK messages.

The tight coupling between CON and ACK also relieves the ACK of the need to echo back information from the request, such as the Token Option supplied by the client. We say that a response carried in an ACK *_pertains_* to the request in the corresponding CON.

2.1.1.2. Asynchronous response

Not all interactions are as simple as the basic synchronous exchange shown. For example, a server might need longer to obtain the representation of the resource requested than it can wait sending back the acknowledgment, without risking the client to repeatedly retransmit the request. To handle this case, the response is decoupled from the transaction layer acknowledgment. Actually, the latter does not carry any message at all.

As the client cannot know that this will be the case, it sends exactly the same confirmable message with the same request. The

server maybe attempts to obtain the resource (e.g., by acting as a proxy) and times out an ACK timer, or it immediately sends an acknowledgment knowing in advance that there will be no quick answer. The acknowledgment effectively is a promise that the request will be acted upon, see Figure 3. Since no Token Option was included in the initial request, an "Token Option required by server (CoAP 240)" error will be returned in the ACK. The client would then repeat the request, now including a Token Option. For a request where an asynchronous response is expected, the Token Option can be included in the first request.

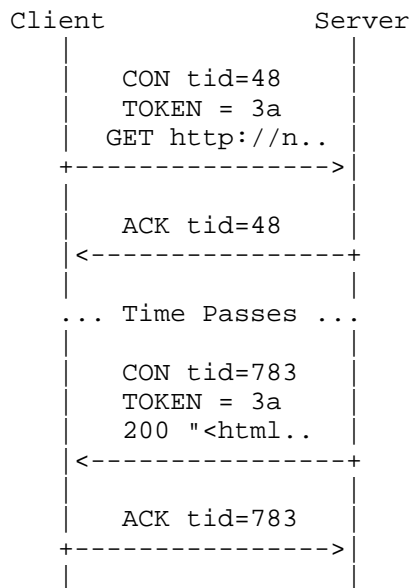


Figure 3: An asynchronous GET transaction

When the server finally has obtained the resource representation and is ready to send the response, it initiates a transaction to the client. This new transaction has its own transaction ID, so there is no automatic coupling of the response to the request. Instead, the Token Option is echoed back to the client in order to associate the response to the original request. To ensure that this message is not lost, it is again sent as a confirmable message and answered by the client with an ACK, citing the new TID chosen by the server.

As a special failure situation, a client may no longer be aware that it sent a request, e.g., if it does not have stable storage and was rebooted in the meantime. This can be indicated by a special "Reset" message, as shown in Figure 4.

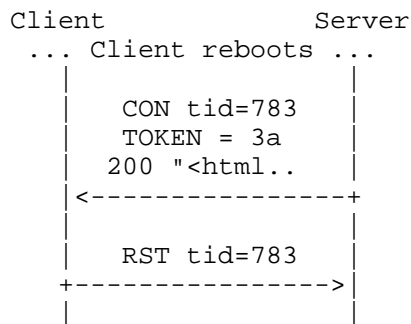


Figure 4: An orphaned transaction

2.2. Transaction messages

The CoAP transactions make use of four different message types, described in this section. These messages are transparent to the request/response carried over them.

2.2.1. Confirmable (CON)

Some messages require an acknowledgment, either just to know they did arrive or also to deliver the reply to a request. We call these messages "Confirmable". When no packets are lost, each Confirmable message elicits exactly one return message of type Acknowledgment or type Reset.

2.2.2. Non-Confirmable (NON)

Some other messages do not require an acknowledgment. This is particularly true for messages that are repeated regularly for application requirements, such as repeated readings from a sensor where eventual arrival is sufficient.

2.2.3. Acknowledgment (ACK)

An Acknowledgment message acknowledges that a specific Confirmable message (identified by its Transaction ID) arrived. As with all of the message types itself, it may carry a payload and some options to provide more details, such as the result of a request that was carried in the Confirmable.

2.2.4. Reset (RST)

A Reset message indicates that a specific Confirmable message was received, but some context is missing to properly process it. This condition is usually caused when the receiving node has rebooted and

has forgotten some state that would be required to interpret the message.

2.2.5. Transaction IDs

The Transaction ID is an unsigned integer kept by a CoAP end-point for all of the CoAP Confirmable or Non-Confirmable messages it sends. Each CoAP end-point keeps a single Transaction ID variable, which is changed each time a new Confirmable or Non-Confirmable message is sent regardless of the destination address or port. The Transaction ID is used to match an Acknowledgment with an outstanding request, for retransmission and to discard duplicate messages. The initial Transaction ID should be randomized. The same Transaction ID MUST NOT be re-used within the potential retransmission window, calculated as $\text{RESPONSE_TIMEOUT} * (2 ^ \text{MAX_RETRANSMIT} - 1)$.

2.3. Methods

CoAP supports the basic methods of GET, POST, PUT, DELETE, which are easily mapped to HTTP. In this section each method is defined along with its behavior. A unicast request with an unknown or unsupported Method Code MUST generate a message with a "405 Method Not Allowed" Response Code.

As CoAP methods manipulate resources, they have the same properties of safe (only retrieval) and idempotent (you can invoke it multiple times with the same effects) as HTTP Section 9.1 [RFC2616]. The GET method is safe, therefore it MUST NOT take any other action on a resource other than retrieval. The GET, PUT and DELETE methods MUST be performed in such a way that they are idempotent. Unlike PUT, POST is not idempotent because the URI in the request indicates the resource that will handle the enclosed body. This resource indicated by the POST may be used for data processing, a gateway to other protocols and it may create a new resource as a result of the POST.

2.3.1. GET

The GET method retrieves the information of the resource identified by the request URI. Upon success a 200 (OK) response SHOULD be sent.

The response to a GET is cacheable if it meets the requirements in Section 5.

2.3.2. POST

The POST method is used to request the server to create a new subordinate resource under the requested parent URI. If a resource has been created on the server, the response SHOULD be 201 (Created)

including the URI of the new resource in a Location Option with any possible status in the message body. If the POST succeeds but does not result in a new resource being created on the server, a 200 (OK) response code SHOULD be returned.

Responses to this method are not cacheable.

2.3.3. PUT

The PUT method requests that the resource identified by the request URI be updated or created with the enclosed message body. If a resource exists at that URI the message body SHOULD be considered a modified version of that resource, and a 200 (OK) response SHOULD be returned. If no resource exists then the server MAY create a new resource with that URI, resulting in a 201 (Created) response. If the resource could not be created or modified, then an appropriate error response code SHOULD be sent.

Responses to this method are not cacheable.

2.3.4. DELETE

The DELETE method requests that the resource identified by the request URI be deleted. The response 200 (OK) SHOULD be sent on success.

Responses to this method are not cacheable.

2.4. Response Codes

CoAP makes use of a subset of HTTP response codes along with some CoAP specific codes as defined in Section 11.1.

2.5. Options

CoAP makes use of compact, extensible Type-Length-Value (TLV) style options. This section explains the processing of CoAP options along with a summary of the main features implemented in options such as URIs and Content-types.

2.5.1. Option Processing

If no options are to be included, the Option Count field is set to 0 below and the Payload (if any) immediately follows the Transaction ID. If options are to be included, the following rules apply. The number of options is placed in the Option Count field. Each option is then placed in order of Type, immediately following the Transaction ID with no padding. Upon reception, unknown options of

class "elective" MUST be silently skipped. Unknown options of class "critical" in a Confirmable MUST cause the return of a response code "Critical Option not supported (CoAP 242)" including a copy of the critical option number in the payload of the response.

2.5.2. URIs

The Universal Resource Identifier (URI) [RFC3986] is an important feature of the web architecture. CoAP supports URIs similarly to HTTP, e.g. `coap://[2001:DB8::101]/s/temp`, where the authority and path identify the resource to be manipulated.

The CoAP header splits the URI up into four parts with the default `coap://` scheme plus Uri-Authority, Uri-Path and Uri-Query CoAP header options. The full URI is reconstructed as follows:

```
( "coap:" )
( "/" Uri-Authority ) only if Uri-Authority is present
( "/" Uri-Path )
( "?" Uri-Query ) only if Uri-Query is present
```

Where reference to an option is replaced by the value of that option if the option is in the header, and the default value for the option if it is not present. The default value for Uri-Authority and Uri-Path is the empty string. Uri-Query has no default value.

CoAP does not support "." or ".." in URIs, IRIs, nor fragment "#" processing. All URI strings in CoAP MUST use the US-ASCII encoding defined in [RFC3986]. When using the Uri-Path Option the leading slash MUST be omitted. Thus the above example `/s/temp` is included in the Uri-Path Option as `s/temp`.

The authority part of a URI is important in determining the correct representation to return on end-points maintaining virtual servers and for intermediate components such as proxies. For this reason it is important that the full URI can be reconstructed when needed. However, at the same time, it is often advantageous for CoAP to elide the Uri-Authority when it is unknown or identical to the IPv6 destination address for efficiency. The following rules apply to processing a CoAP request:

1. If the Uri-Authority option is absent and the remainder of the URI uniquely identifies a resource the server MAY proceed to execute the request.

2. If an origin server is able to determine the IP destination address of the request, it MAY assume this as the authority of the URI.
3. If no authority can be determined and the server requires the authority to identify the resource it MUST reject the request with "Uri-Authority Option required by server (CoAP 241)".

Application designers are encouraged to make use of short, but descriptive URIs. For example URIs 14 or less bytes in length fit in a more compact option header. In addition, very short URIs such as "/1" can be assigned as an alternative short URI for a resource by the application. The CoRE Link Format includes an attribute to indicate if a short alternative URI of a resource is available [I-D.ietf-core-link-format].

The CoAP protocol scheme is identified in URIs with "coap://" [IANA_TBD_SCHEME].

2.5.3. Content-type encoding

In order to support heterogeneous uses, CoAP is transparent to the use of different application payloads. In order for the application process receiving a packet to properly parse a payload, its content-type should be explicitly known from the header (as e.g. with HTTP). The use of typical binary encodings for XML is discussed in [I-D.shelby-6lowapp-encoding].

String names of Internet media types (MIME types) [RFC2046] are not optimal for use in the CoAP header. Instead, CoAP simply assigns identifiers to a subset of common media and content transfer encoding types. The content-type identifier is optionally included in the Content-type Option Header of messages to indicate the type of the message body. CoAP Content-type identifiers are defined in Section 11.2. In the absence of the Content-type Option the MIME type "text/plain" MUST BE assumed.

3. Message Formats

CoAP makes use of asynchronous transactions using a simple binary header format. This base header may be followed by options in Type-Length-Value (TLV) format. CoAP is bound to UDP as described in Section 4.

Any bytes after the headers in the packet are considered the message payload, if any. The length of the message payload is implied by the datagram length. See Section 4 for further message length

requirements.

3.1. CoAP header

This section defines the CoAP header, which is shared for all CoAP messages. CoAP makes use of an asynchronous transaction model. These transactions are used to carry request/response exchanges, either using a Method Code (GET/PUT/POST/DELETE) to invoke interaction with a resource, or a Response Code carried in an immediate or asynchronous response.

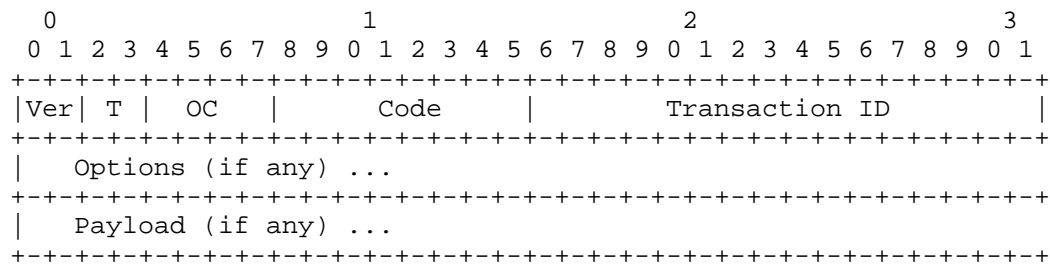


Figure 5: CoAP header format

Header Fields:

Ver: Version. 2-bit unsigned integer. Indicates the version of CoAP. Implementations of this specification **MUST** set this field to 1. Other values are reserved for future versions.

T: 2-bit unsigned integer Transaction Type field. Indicates if this message is Confirmable (0), Non-Confirmable (1), Acknowledgment (2) or Reset (3).

OC: 4-bit unsigned integer Option Count field. Indicates if there are Option Headers following the base header. If set to 0 the payload (if any) immediately follows the base header. If greater than zero the field indicates the number of options to immediately follow the header.

Code: 8-bit unsigned integer. This field indicates the Method or Response Code of a message. The value 0 indicates no code. The values 1-10 are used for Method Codes as defined in Table 1. The values 11-39 are reserved for future use. The values 40-255 are used for Response Codes as defined in Section 11.1.

Transaction ID: 16-bit unsigned integer. A unique Transaction ID assigned by the source and used to match responses. The Transaction ID MUST be changed for each new request (regardless of the end-point) and MUST NOT be changed when retransmitting a request (see Section 2.2.5).

Method	Code
GET	1
POST	2
PUT	3
DELETE	4

Table 1: Method Codes

3.2. Header options

CoAP messages may also include one or more header options in TLV format. Options MUST appear in order of option type (see Table 2). A delta encoding is used between each option header, with the Type identifier for each Option calculated as the sum of its Option Delta field and the Type identifier of the preceding Option in the message, if any, or zero otherwise.

Each option header includes a Length field which can be extended by an octet for options with values longer than 14 octets. CoAP options include the concept of Critical (odd value) and Elective (even value) options (see Section 2.5.1).

Each option has the following format:

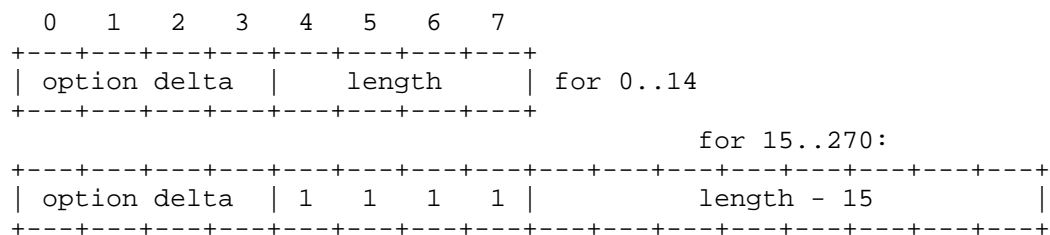


Figure 6: Header option format

Option delta: 4-bit unsigned integer. This field defines the difference between the option Type of this option and the previous option (or zero for the first option). In other words, the Type identifier is calculated by simply summing the Option delta fields of this and previous options before it. The Option Values 14, 28, ... are reserved for no-op options with no value (they are ignored) and are used for deltas larger than 14. Thus these can be used as "fenceposts" if deltas larger than 15 would otherwise be required.

Length: Length Field. Normally Length is a 4-bit unsigned integer allowing values of 0-14 octets. When the length is 15 or more, another byte is added as an 8-bit unsigned integer plus 15 allowing values of 15-270 octets.

Option Value The value in the format defined for that option in Table 2 of Length octets. Options MAY use variable length values.

The following options are defined in this document. The Default column indicates the value to be assumed in the absence of this option (if any).

Type	C/E	Name	Data type	Length	Default
0	-	Reserved	-	-	-
1	C	Content-type	8-bit unsigned integer	1 B	0 (text/plain)
2	E	Max-age	Variable length unsigned integer	1-4 B	60 seconds
3	C	-	Reserved	-	-
4	E	Etag	Sequence of bytes	1-4 B	-
5	C	Uri-Authority	String	1-270 B	" "
6	E	Location	String	1-270 B	-
7	-	-	-	-	-
9	C	Uri-Path	String	1-270 B	" "
10	E	Subscription lifetime [I-D.ietf-core-observe]	Variable length unsigned integer	0-4 B	0

11	C	Token	Sequence of bytes	1-2 B	-
13	C	Block [I-D.ietf-core-block]	Unsigned Integer	1-3 B	0
15	C	Uri-Query	String	1-270 B	-

Table 2: Option headers

3.2.1. Content-type Option

The Content-type Identifier Option indicates the Internet media type identifier of the message-body, see Section 11.2 for the encoding and identifier tables. A Content-type Identifier Option SHOULD be included if there is a payload included with a CoAP message. In the absence of the Content-type Option the MIME type "text/plain" (0) MUST be assumed. This option MUST be supported by all end-points. This option MUST NOT occur more than once in a header.

3.2.2. Uri-Authority Option

The Uri-Authority Option indicates the authority (host + port) part of a URI, and conforms to "host [: port]" (section 3.2 of [RFC3986], excluding use of userinfo) Examples of this option include "[2001:DB8::101]", "198.51.100.0:8000" and "sensor.example.com". This option is used by servers to determine which resource to return and by intermediate components, e.g. when accessing a resource via a proxy. Section 2.5.2 specifies the rules for URIs in CoAP. This option SHOULD be included in a request when the authority of the URI is known. This option MUST be supported by an end-point implementing proxy functionality. This option MUST NOT occur more than once in a header.

3.2.3. Uri-Path Option

The Uri-Path Option indicates the absolute path part of a URI, and conforms to the path-noscheme (path-rootless?) rule in section 3.3 of [RFC3986]. One example of an absolute path in this option is "s/light". In the absence of this option, the path is assumed to be "/". Section 2.5.2 specifies the rules for URIs in CoAP. The leading slash is assumed and MUST be omitted. This option MUST be supported by all end-points. This option MUST NOT occur more than once in a header.

3.2.4. Uri-Query Option

The Uri-Query Option indicates the query part of a URI (if any), and conforms to the query rule in section 3.4 of [RFC3986]. In the absence of this option, there is assumed to be no query string. Section 2.5.2 specifies the rules for URIs in CoAP. This option **MUST** be supported by all end-points. This option **MUST NOT** occur more than once in a header.

3.2.5. Location Option

The Location Option indicates the location of a resource as an absolute path URI and is similar to the Uri-Path Option. The Location Option **MAY** be included in a response to indicate the Location of a new resource created with POST or together with a 30x response code. The leading slash is assumed and **MUST** be omitted. This option **MUST NOT** occur more than once in a header.

3.2.6. Max-age Option

The Max-age Option indicates the maximum age of the resource for use in cache control in seconds. The option value is represented as a variable length unsigned integer between 8 and 32 bits. A default value of 60 seconds is assumed in the absence of this option.

When included in a request, Max-age indicates the maximum age of a cached representation of that resource the client will accept. When included in a response, Max-age indicates the maximum time the representation may be cached before it **MUST** be discarded. This option **MUST NOT** occur more than once in a header.

3.2.7. Etag Option

The Etag Option is an opaque sequence of bytes which specifies the version of a resource representation. An Etag may be generated for a resource in any number of ways including a version, checksum, hash or time. An end-point receiving an Etag **MUST** treat it as opaque and make no assumptions about its format. The Etag **MAY** be included in a response to indicate to a client if a resource has changed. The Etag **SHOULD** be included in a request used for a cache refresh to indicate the client's current version of the resource (see Section 5.2).

3.2.8. Token Option

The Token Option is an opaque sequence of 1-2 bytes which is used to match a request with a response and is meant for use with asynchronous responses by this specification. The Token is generated by a client and included in a way that Token values currently in use

are unique. Tokens have the following rules:

If a Token Option is included in a request, the response (and any subsequent delayed responses) MUST include the same value in a Token Option.

If a Token Option is included in a request, any resulting delayed response SHOULD NOT include the URI option (for sake of efficiency) as the Token is sufficient for matching it with the request.

If a request does not include a Token option, the server MUST provide its ReST response within the transaction response. If it cannot do so (i.e., can only satisfy the request through an asynchronous response), it MUST respond with error "Token Option required by server (CoAP 240)".

This option MUST NOT occur more than once in a header.

4. UDP Binding

The CoAP protocol operates by default over UDP. CoAP may also be used with Datagram Transport Layer Security (DTLS) as described in Section 10. CoAP could also be used over other transports such as TCP or SCTP, the specification of which is out of this document's scope.

The goal of binding CoAP to UDP is to provide the bare minimum features for the protocol to operate over UDP, without trying to re-create the full feature set of a transport like TCP. CoAP over UDP has the following features:

- o Simple stop-and-wait retransmission reliability with exponential back-off as described in Section 4.2 for Confirmable messages.
- o Transaction ID for response matching as described in Section 2.2.5.
- o Multicast support as described in Section 4.1.

The length of the Payload in a CoAP message is calculated from the datagram length. While specific link layers make it beneficial to keep CoAP messages small enough to fit into their link layer packets (see Section 1), this is a matter of implementation quality. The CoAP specification itself provides only an upper bound to the message size. A CoAP message SHOULD fit within a single IP packet and MUST fit within a single IP datagram. If the Path MTU is not known for a

destination, an MTU of 1280 octets SHOULD be assumed.

4.1. Multicast

CoAP supports the use of multicast destination addresses. Multicast messages SHOULD be Non-Confirmable. If a Confirmable multicast message is sent then retransmission MUST NOT be performed. Furthermore, a destination end-point to a multicast Confirmable message MUST only send an Acknowledgment if the response code included indicates success (Code = 2XX) in order to eliminate error code response floods. Other mechanisms for avoiding congestion from multicast requests are being considered in [I-D.eggert-core-congestion-control].

4.2. Retransmission

A CoAP end-point keeps track of open Confirmable messages it sent that are waiting for a response. Each entry includes at least the destination IP address and port of the original message, the message itself, a retransmission counter and a timeout. When a Confirmable is sent, an entry is made for that message with a default initial timeout of RESPONSE_TIMEOUT and the retransmission counter set to 0. When a matching Acknowledgment is received for an entry, the entry is invalidated. When a timeout is triggered for an entry and the retransmission counter is less than MAX_RETRANSMIT, the original message is retransmitted to the destination without modification, the retransmission counter is incremented, and the timeout is doubled. If the retransmission counter reaches MAX_RETRANSMIT on a timeout, then the entry is removed and the application process informed of delivery failure.

For CoAP messages sent to IP multicast addresses, retransmission MUST NOT be performed. Therefore MAX_RETRANSMIT is always set to 0 when the destination address is multicast.

4.3. Congestion Control

In addition to the exponential back-off mechanism in Section 4.2, further congestion control optimizations are being considered and tested for CoAP. These congestion control mechanism under consideration are described in [I-D.eggert-core-congestion-control].

4.4. Default Port

The CoAP default port number [IANA_TBD_PORT] MUST be supported by a server for resource discovery (see Section 6) and SHOULD be supported for providing access to other resources. In addition other end-points may be hosted in the dynamic port space.

When a CoAP server is hosted by a 6LoWPAN node, it SHOULD also support a port in the 61616-61631 compressed UDP port space defined in [RFC4944]. The specific port number in use will be communicated using e.g. CoRE discovery [I-D.ietf-core-link-format].

5. Caching

CoAP end-points are by definition constrained by bandwidth and processing power. To optimize the performance of data transfer under these constraints, we use caching features consistent with HTTP. Caching includes the following concepts:

- o Cache life of a resource is controlled via the Max-Age header option
- o Cache refresh and versioning of a resource is controlled via the Etag header option
- o Proxies between a client and end-point may participate in the caching process on behalf of sleeping end-points and to avoid unnecessary traffic on the constrained network

5.1. Cache control

When an end-point responds to a GET request by sending a representation of the resource, it SHOULD specify the Max-Age header option. The Max-Age specifies the cache life of the resource in seconds. Resources which change rapidly will have a short cache life, and resources which change infrequently should specify a long cache life. If Max-Age is unspecified in a GET response, then it is assumed to be 60 seconds. If an end-point wishes to disable caching, it must explicitly specify a Max-Age of zero seconds.

When a client reads the response from a GET request, it should cache the resource representation for the cache lifetime as specified by the Max-Age header. During the cache lifetime, the client SHOULD use its cached version and avoid performing additional GETs for the resource.

In general, the origin server end-point is responsible for determining cache age. However, in some cases a client may wish to determine its own tolerance for cache staleness. In this case, a client may specify the Max-Age header during a GET request. If the client's Max-Age is of a shorter duration than the age of a cached resource, then the proxy or end-point SHOULD perform a cache refresh. If the client specifies a Max-Age of zero seconds, then the response MUST discard the cached representation and return a fresh

representation.

5.2. Cache refresh

After the expiration of the cache lifetime, clients and proxies can refresh their cached representation of a resource. Cache refresh is accomplished using a GET request which will return a representation of the resource's current state.

If the end-point has the capability to version the resource, then the end-point should include the Etag header option in the response to a GET request. The Etag is a variable length sequence of bytes which captures a version identifier of the resource. The Etag is an opaque identifier; clients MUST NOT infer any semantics from the Etag value.

If an end-point specifies the Etag header option with a response, then the client SHOULD specify a matching Etag header option in their GET request during cache refresh. If the end-point's version of the resource is unmodified, then the server SHOULD return a 304 response with no payload to avoid retransmitting the resource representation.

5.3. Proxying

A proxy is defined as a CoAP end-point which services cached requests on behalf of other CoAP end-points. Any node in a CoAP network may act as a proxy, although in general the node between the constrained network and the Internet at large SHOULD always support proxy functionality.

Proxies should be used under the following scenarios:

- o Clients external to the constrained network SHOULD always make requests through a proxy to limit traffic on the constrained network
- o Clients internal to the constrained network MAY use a proxy based on network topology when performance warrants
- o Clients of sleeping devices MUST use a proxy to access resources while the device is sleeping

Proxy requests are made as normal CON requests to the proxy end-point. All proxy requests MUST use the Uri-Authority header to indicate the origin server's IP address. The host part is case insensitive and may be an IPv4 literal, IPv6 literal in square brackets, or a registered name. The port number is optional, if omitted or zero-length it is assumed to be the default CoAP port (see Section 4.4).

When a request is made to a proxy, then the following steps are taken:

1. If the authority (host and port) is recognized as identifying the proxy end-point, then the request MUST be treated as a local request and the path part is used as Uri-Path
2. If the proxy does not contain a fresh cached representation of the resource, then the proxy MUST attempt to refresh its cache according to section 5.2. The origin server's IP address and port is determined by the authority part of the full URI. The Uri-Path option for the refresh request is determined by the path part of the full URI.
3. If the proxy fails to obtain a fresh cached representation, then a 502 Bad Gateway error code MUST be returned
4. The proxy returns the cached representation on behalf of the origin server

All CoAP options are considered end-to-end and MUST be stored as part of the cache entry and MUST be transmitted in the proxy's response. The Max-Age option should be adjusted by the proxy for each response using the formula: $\text{proxy-max-age} = \text{original-max-age} - \text{cache-age}$. For example if a request is made to a proxied resource that was refreshed 20sec ago and had an original Max-Age of 60sec, then that resource's proxied Max-Age is now 40sec.

6. Resource Discovery

The discovery of resources offered by a CoAP end-point is extremely important in machine-to-machine applications where there are no humans in the loop and static interfaces result in fragility. A CoAP end-point SHOULD support the CoRE Link Format of discoverable resources as described in [I-D.ietf-core-link-format].

7. HTTP Mapping

CoAP supports a limited subset of HTTP functionality, and thus a mapping to HTTP is straightforward. There might be several reasons for mapping between CoAP and HTTP, for example when designing a web interface for use over either protocol or when realizing a CoAP-HTTP proxy. Likewise, CoAP could equally be mapped to other protocols such as XMPP or SIP, the definition of which is out of scope.

The mapping of CoAP to HTTP is a straightforward conversion of the

CoAP method or response code, content-type and options to the corresponding HTTP feature. The payload is carried in an equivalent way by both protocols. The mapping of HTTP to CoAP requires checking for methods, response codes, options and content-types that are not supported by CoAP. A mapping SHOULD attempt to map options, response codes and content-types to a suitable alternative if possible. Otherwise the unsupported feature SHOULD be silently dropped if possible, or an appropriate error code generated otherwise.

The caching and proxying of CoAP is specified in Section 5. In a similar manner, caching and proxying MAY be performed between CoAP and HTTP by an intermediate node. A proxy SHOULD respond with a 502 (Bad Gateway) error to HTTP requests which can not be successfully mapped to CoAP. CoAP transaction messages are transparent to request/response exchanges and MUST have no affect on a proxy function.

8. Protocol Constants

This section defines the relevant protocol constants defined in this document:

RESPONSE_TIMEOUT	1 second
MAX_RETRANSMIT	5

9. Examples

Figure 7 shows a basic request sequence. A client makes a Confirmable GET request for the resource /temperature to the server with a Transaction ID of 1234. The request includes one Uri-Path Option (delta 0 + 9 = 9) "temperature" of Len = 11. This request is a total of 16 octets long. The corresponding Acknowledgment is of Code 200 OK and includes a Payload of "22.3 C". The Transaction ID is 1234, thus the transaction is successfully completed. The response is 10 octets long and a Content-type of 0 (text/plain) is assumed as there is no Content-type Option.

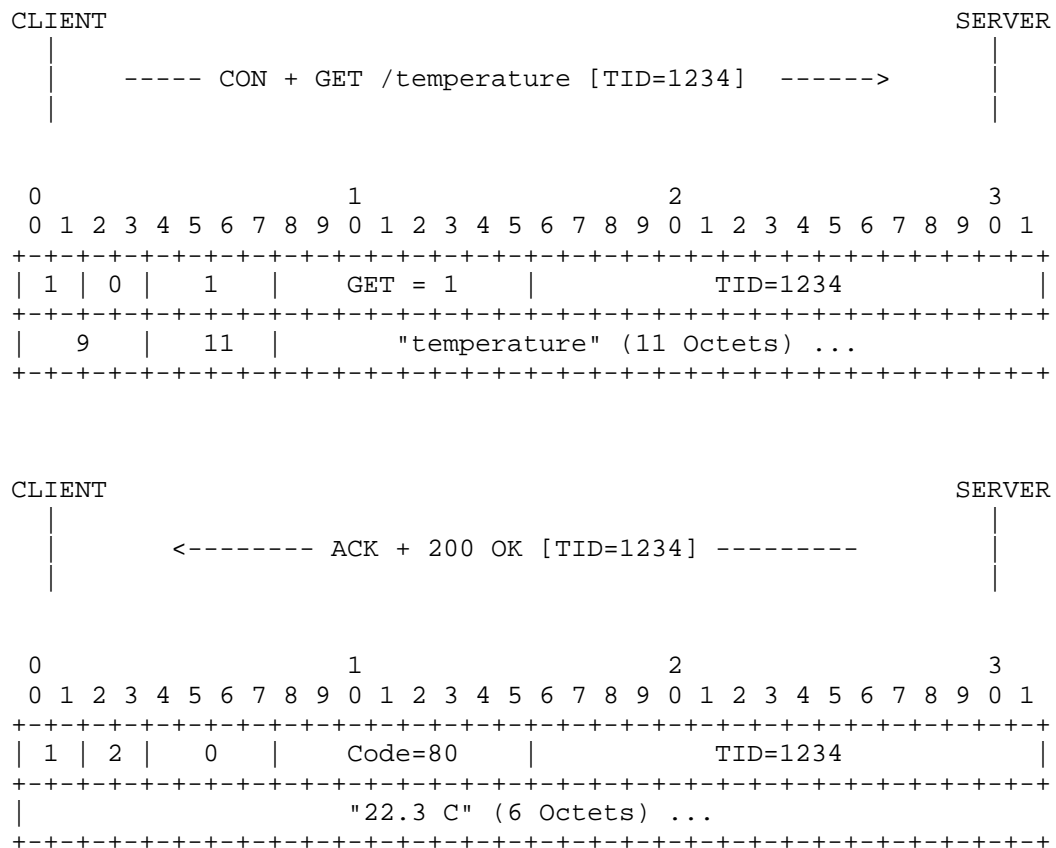


Figure 7: Basic request/response

Figure 8 shows an example of a retransmission using the previous request. The first ACK from the server is lost, and after RESPONSE_TIMEOUT seconds the client retransmits the request.

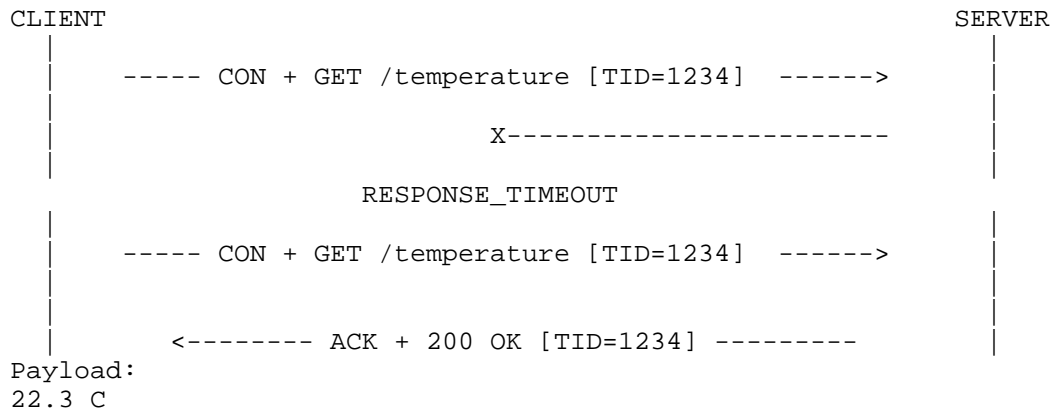


Figure 8: Basic request/response

Figure 9 shows an example of resource discovery. Here a unicast GET request is made to the server for /.well-known/core [I-D.ietf-core-link-format], which returns a list of two resource descriptions. The client then decides to make a request for the short URI of /sensor/light (/l). Requesting /sensors/light would result in the same representation.

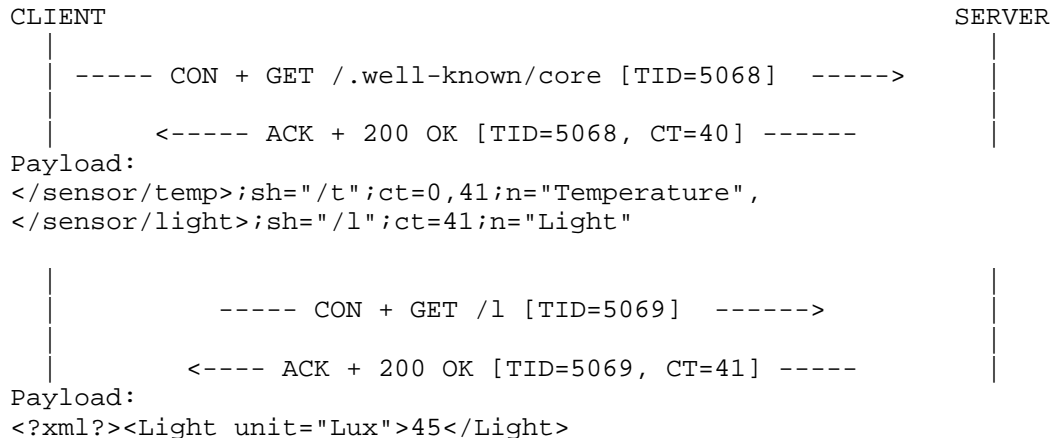


Figure 9: Basic request/response

Figure 10 shows an example of a multicast request. Here a client sends a request for /.well-known/core with a query for ?n=Light

(Resource name = Light) to all-nodes link-scope multicast. There are 3 servers on the link: A, B and C. Servers A and B have a matching resource, therefore they send back a successful 200 OK response with the matching resource in the payload. C does not attempt to send a response.

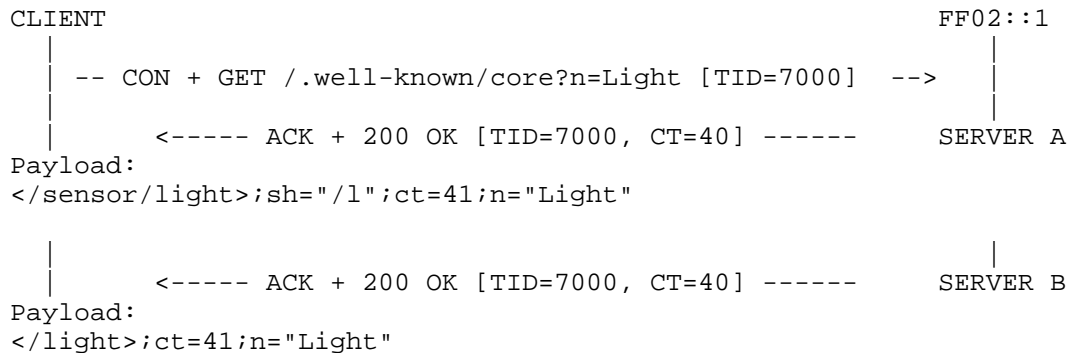


Figure 10: Basic request/response

10. Security Considerations

This section describes mechanisms that can be used to secure CoAP and analyzes the possible threats to the protocol and its limitations. Security bootstrapping (authenticating nodes and setting up keys) in constrained environments is considered in [I-D.oflynn-core-bootstrapping].

During the bootstrap and enrollment phases, a CoAP device is provided with the key security information that it needs. How this is done is out of scope for this specification but a couple ways of doing this are described in [I-D.oflynn-core-bootstrapping]. At the end of the enrollment and bootstrap, the device will be in one of four security modes with the following information for the given mode:

NoSec: There is no protocol level security.

SharedKey: There is one shared key between all the nodes that this CoAP nodes needs to communicate with.

MultiKey: There is a list of shared keys and each key includes a list of which nodes it can be used to communicate with. At the extreme there may be one key for each node this CoAP node needs to communicate with.

Certificate: The device has an asymmetric key pair with a X.509 [RFC5280] certificate that binds it to its Authority Name and is signed by a some common trust root. The device also has a list or root trust anchors that can be used for validating a certificate. There may be an optional shared key that all the nodes that communicate have access too.

The Authority Name in the certificate is the name that would be used in the Authority part of a CoAP URI. It is worth noting that this would typically not be either an IP address or DNS name but would instead be a long term unique identifier for the device such as the EUI-64. The discovery process used in the system would build up the mapping between IP addresses of the given devices and the Authority Name for each device. Some devices could have more than one Authority and would need more than a single certificate.

In the "NoSec" mode, the system simply sends the packets over normal UDP over IP. The system is secured only by physically keeping attackers from being able to send or receive packets from the network with the CoAP nodes. The other three security modes can be achieved with IPsec or DTLS.

10.1. Securing CoAP with IPsec

One mechanism to secure CoAP in constrained environments is the IPsec Encapsulating Security Payload (ESP) [RFC4303]. Using IPsec ESP with the appropriate configuration, it is possible for many constrained devices to support encryption with built-in link-layer encryption hardware. For example, many IEEE 802.15.4 radio chips are compatible with AES-CBC (with 128-bit keys) [RFC3602] as defined for use with IPsec in [RFC4835]. Alternatively, and to avoid the need for padding, nodes could directly use the more widely supported AES-CCM as defined for use with IPsec in [RFC4309], if the security considerations in section 9 of that specification can be fulfilled.

When using IPsec to secure CoAP, both authentication and confidentiality SHOULD be applied as recommended in [RFC4303]. The use of IPsec between CoAP end-points is transparent to the application layer and does not require special consideration for a CoAP implementation.

IPsec may not be appropriate for all environments. For example, IPsec support is not available for many embedded IP stacks and even

in full PC operating systems or on backend web servers, application developers may not have sufficient access to configure or enable IPsec or to add a security gateway to the infrastructure. Problems with firewalls and NATs may furthermore limit the use of IPsec.

10.2. Securing CoAP with DTLS

Just as HTTP may be secured using Transport Layer Security (TLS) over TCP, CoAP may be secured using Datagram TLS (DTLS) [RFC4347] over UDP. This section describes how to secure CoAP with DTLS, along with the minimal configurations appropriate for constrained environments. DTLS is in practice TLS with added features to deal with the unreliable nature of the UDP transport.

In some constrained nodes (limited flash and/or RAM) and networks (limited bandwidth or high scalability requirements) DTLS may not be applicable. The protocol is an order of magnitude more complex than CoAP and has appreciable handshake overhead needed to maintain security sessions. DTLS makes sense for applications where the session maintenance makes it compatible with application flows and sufficient resources are available on the constrained nodes and for the added network overhead.

Devices SHOULD support the Server Name Indication (SNI) to indicate their Authority Name in the SNI HostName field as defined in Section 3 of draft-ietf-tls-rfc4366-bis. This is needed so that when a host that acts as a virtual server for multiple Authorities receives a new DTLS connection, it knows which keys to use for the DTLS session.

DTLS connections with certificates are set up using mutual authentication so they can remain up and be reused for future transaction in either direction. Devices can close a DTLS connection when they need to recover resources but in general they should keep the connection up for as long as possible. Closing the DTLS connection after every CoAP transaction is very inefficient.

10.2.1. SharedKey & MultiKey Modes

When forming a connection to a new node, the system selects an appropriate key based on which nodes it is trying to reach then forms a DTLS session using a PSK (Pre-Shared Key) mode of DTLS. Implementations SHOULD support the mandatory to implement cipher suite TLS_PSK_WITH_AES_128_CBC_SHA as specified in [RFC5246].

10.2.2. Certificate Mode

As with IPsec, DTLS should be configured with a cypher suite compatible with any possible hardware engine on the node, for example

AES-CBC in the case of IEEE 802.15.4. Implementations SHOULD support the mandatory to implement cipher suite TLS_RSA_WITH_AES_128_CBC_SHA as specified in [RFC5246].

When a new connection is formed, the certificate from the remote device needs to be verified. If the CoAP node has a source of absolute time, then the node SHOULD check the validity dates are of the certificate are within range. The certificate MUST also be signed by an appropriate chain of trust. If the certificate contains a SubjectAltName, then the Authority Name MUST match at least one of the authority names of any CoAP URI found in a URI type fields in the SubjectAltName set. If there is no SubjectAltName in the certificate, then the Authoritative Name must match the CN found in the certificate using the matching rules defined in [RFC2818] with the exception that certificates with wildcards are not allowed.

If the system has a shared key in addition to the certificate, then a cipher suite that includes the shared key such as TLS_RSA_PSK_WITH_AES_128_CBC_SHA SHOULD be used.

10.3. Threat analysis and protocol limitations

This section is meant to inform protocol and application developers about the security limitations of CoAP as described in this document. As CoAP realizes a subset of the features in HTTP/1.1, the security considerations in Section 15 of [RFC2616] are also pertinent to CoAP. This section concentrates on describing limitations specific to CoAP and CoRE.

10.3.1. Processing URIs

Complex parsers are known as a likely source of vulnerabilities, such as the ability to remotely crash a node, or even remotely execute arbitrary code on it. The URI processing code in CoAP implementations should be subjected to stringent tests with various forms of malformed parameters. (See also section 15.2 of [RFC2616].)

10.3.2. Proxying and Caching

As mentioned in 15.2 of [RFC2616], which see, proxies are by their very nature men-in-the-middle, breaking any IPsec or DTLS protection that a direct CoAP transaction might have. They are therefore interesting targets for breaking confidentiality or integrity of CoAP transactions. As noted in [RFC2616], they are also interesting targets for breaking availability.

The threat to confidentiality and integrity of transaction data is amplified where proxies also cache. Note that CoAP does not define

any of the cache-suppressing Cache-Control options that HTTP/1.1 provides to better protect sensitive data.

Finally, a proxy that fans out asynchronous responses to multiple original requesters may provide additional amplification (see below).

10.3.3. Risk of amplification

CoAP servers generally reply to a request packet with a response packet. This response packet may be significantly larger than the request packet. An attacker might use CoAP nodes to turn a small attack packet into a larger attack packet, an approach known as amplification. There is therefore a danger that CoAP nodes could become implicated in denial of service (DoS) attacks by using the amplifying properties of the protocol: An attacker that is attempting to overload a victim but is limited in the amount of traffic it can generate, can use amplification to generate a larger amount of traffic.

This is particularly a problem in nodes that enable NoSec access and that are accessible from an attacker and can access potential victims (e.g. on the general Internet), as the UDP protocol provides no way to verify the source address given in the request packet. An attacker need only place the IP address of the victim in the source address of a suitable request packet to generate a larger packet directed at the victim.

As a mitigating factor, many constrained network will only be able to generate a small amount of traffic, which may make CoAP nodes less attractive for this attack. However, the limited capacity of the constrained network makes the network itself a likely victim of an amplification attack.

A CoAP server can reduce the amount of amplification it provides to an attacker by using slicing/blocking modes of CoAP [I-D.ietf-core-block] and offering large resource representations only in relatively small slices. E.g., for a 1000 byte resource, a 10-byte request might result in an 80-byte response (with a 64-byte block) instead of a 1016-byte response, considerably reducing the amplification provided.

CoAP supports also the use of multicast IP addresses in requests, an important requirement for M2M. Multicast CoAP requests may be the source of accidental or deliberate denial of service attacks, especially over constrained networks. This specification attempts to reduce the amplification effects of multicast requests by limiting when a response is returned. To limit the possibility of malicious use, CoAP servers SHOULD NOT accept multicast requests that can not

be authenticated. If possible a CoAP server SHOULD limit the support for multicast requests to specific resources where the feature is required.

On some general purpose operating systems providing a Posix-style API, it is not straightforward to find out whether a packet received was addressed to a multicast address. While many implementations will know whether they have joined a multicast group, this creates a problem for packets addressed to multicast addresses of the form FF0x::1, which are received by every IPv6 node. Implementations SHOULD make use of modern APIs such as IPV6_RECVPKTINFO [RFC3542], if available, to make this determination.

11. IANA Considerations

[IANA_TBD_SCHEME] This document suggests the scheme coap:// to identify this protocol in a URI. The string "coap" should similarly be used in well-known port and service discovery registrations.

[IANA_TBD_PORT] Apply for a well-known port number in the 0-1023 space as CoAP end-points are usually executed by an operating system or root process. <http://www.iana.org/assignments/port-numbers>

[IANA_TBD_MIME] A new registry is required for the Internet MIME type identifier space for CoAP as described in Section 11.2.

11.1. Codes

CoAP makes use of (a subset of) the HTTP status codes defined in [RFC2616] plus some CoAP-specific status codes. The HTTP status code is encoded into an 8-bit unsigned integer code with the mapping defined in Table 3. The use of these codes is defined throughout this document using the HTTP Name (except for CoAP-specific codes).

Code	HTTP Name
40	100 Continue
80	200 OK
81	201 Created
124	304 Not Modified
160	400 Bad Request
164	404 Not Found
165	405 Method Not Allowed
175	415 Unsupported Media Type
200	500 Internal Server Error
202	502 Bad Gateway
203	503 Service Unavailable
204	504 Gateway Timeout
240	Token Option required by server
241	Uri-Authority Option required by server
242	Critical Option not supported

Table 3: CoAP Codes

11.2. Content Types

Internet media types are identified by a string in HTTP, such as "application/xml". This string is made up of a top-level type "application" and a sub-type "xml" [RFC2046]. In order to minimize the overhead of using these media types to indicate the type of message payload, CoAP defines an identifier encoding scheme for a subset of Internet media types. It is expected that this table of identifiers will be extensible and maintained by IANA for values of 0-200 [IANA_TBD_MIME].

The Content-type Option is formatted as an 8-bit unsigned integer. Initial mappings from Internet media types to a suitable identifier is shown in Table 4. Composite high-level types (multipart and message) are not supported. Identifier values from 201-255 are reserved for vendor specific, application specific or experimental use and are not maintained by IANA.

Internet media type	Identifier
text/plain (UTF-8)	0
text/xml (UTF-8)	1
text/csv (UTF-8)	2
text/html (UTF-8)	3
image/gif	21
image/jpeg	22
image/png	23
image/tiff	24
audio/raw	25
video/raw	26
application/link-format [I-D.ietf-core-link-format]	40
application/xml	41
application/octet-stream	42
application/rdf+xml	43
application/soap+xml	44
application/atom+xml	45
application/xmpp+xml	46
application/exi	47
application/x-bxml	48
application/fastinfoset	49
application/soap+fastinfoset	50
application/json	51

Table 4: Media type identifiers

12. Acknowledgments

Special thanks to Carsten Bormann, Klaus Hartke, Peter Bigot and Cullen Jennings for substantial contributions to the ideas and text in the document (Section 2.1.1, Section 2.1.2, Section 2.2, Section 3.2, Section 10), along with countless detailed reviews and discussions.

Thanks to Michael Stuber, Richard Kelsey, Guido Moritz, Peter Van Der Stok, Adriano Pezzuto, Lisa Dussealt, Alexey Melnikov, Gilbert Clark, Salvatore Loreto, Petri Mutka, Szymon Sasin, Robert Quattlebaum, Robert Cragie, Angelo Castellani, Tom Herbst, Ed Beroeset, Gilman Tolle, Robby Simpson, Colin O'Flynn and David Ryan for helpful comments and discussions that have shaped the document.

13. Changelog

Changes from ietf-02 to ietf-03:

- o Token Option and related use in asynchronous requests added (#25).
- o CoAP specific error codes added (#26).
- o Erroring out on unknown critical options changed to a MUST (#27).
- o Uri-Query option added.
- o Terminology and definitions of URIs improved.
- o Security section completed (#22).

Changes from ietf-01 to ietf-02:

- o Sending an error on a critical option clarified (#18).
- o Clarification on behavior of PUT and idempotent operations (#19).
- o Use of Uri-Authority clarified along with server processing rules. Uri-Scheme option removed. (#20, #23)
- o Resource discovery section removed to a separate CoRE Link Format draft (#21)
- o Initial security section outline added.

Changes from ietf-00 to ietf-01:

- o New cleaner transaction message model and header (#5)
- o Removed subscription while being designed (#1)
- o Section 2 re-written (#3)
- o Text added about use of short URIs (#4)
- o Improved header option scheme (#5, #14)
- o Date option removed while being designed (#6)

- o New text for CoAP default port (#7)
- o Completed proxying section (#8)
- o Completed resource discovery section (#9)
- o Completed HTTP mapping section (#10)
- o Several new examples added (#11)
- o URI split into 3 options (#12)
- o MIME type defined for link-format (#13, #16)
- o New text on maximum message size (#15)
- o Location Option added

Changes from shelby-01 to ietf-00:

- o Removed the TCP binding section, left open for the future.
- o Fixed a bug in the example.
- o Marked current Sub/Notify as (Experimental) while under WG discussion.
- o Fixed maximum datagram size to 1280 for both IPv4 and IPv6 (for CoAP-CoAP proxying to work).
- o Temporarily removed the Magic Byte header as TCP is no longer included as a binding.
- o Removed the Uri-code Option as different URI encoding schemes are being discussed.
- o Changed the rel= field to desc= for resource discovery.
- o Changed the maximum message size to 1024 bytes to allow for IP/UDP headers.
- o Made the URI slash optimization and method impotence MUSTs
- o Minor editing and bug fixing.

Changes from shelby-00 to shelby-01:

- o Unified the message header and added a notify message type.
- o Renamed methods with HTTP names and removed the NOTIFY method.
- o Added a number of options field to the header.
- o Combines the Option Type and Length into an 8-bit field.
- o Added the magic byte header.
- o Added new Etag option.
- o Added new Date option.
- o Added new Subscription option.
- o Completed the HTTP Code - CoAP Code mapping table appendix.
- o Completed the Content-type Identifier appendix and tables.
- o Added more simplifications for URI support.
- o Initial subscription and discovery sections.
- o A Flag requirements simplified.

14. References

14.1. Normative References

- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [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.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, April 2007.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

14.2. Informative References

- [I-D.eggert-core-congestion-control]
Eggert, L., "Congestion Control for the Constrained Application Protocol (CoAP)",
draft-eggert-core-congestion-control-00 (work in progress), June 2010.
- [I-D.ietf-core-block]
Shelby, Z. and C. Bormann, "Blockwise transfers in CoAP",
draft-ietf-core-block-00 (work in progress), October 2010.
- [I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format",
draft-ietf-core-link-format-01 (work in progress),
October 2010.
- [I-D.ietf-core-observe]
Hartke, K. and Z. Shelby, "Observing Resources in CoAP",
draft-ietf-core-observe-00 (work in progress),
October 2010.
- [I-D.oflynn-core-bootstrapping]
Sarikaya, B. and R. Cragie, "Initial Configuration of Resource-Constrained Devices",
draft-oflynn-core-bootstrapping-01 (work in progress),

July 2010.

- [I-D.shelby-6lowapp-encoding]
Shelby, Z., Luimula, M., and D. Peintner, "Efficient XML Encoding and 6LowApp", draft-shelby-6lowapp-encoding-00 (work in progress), October 2009.
- [I-D.shelby-core-coap-req]
Shelby, Z., Stuber, M., Sturek, D., Frank, B., and R. Kelsey, "CoAP Requirements and Features", draft-shelby-core-coap-req-01 (work in progress), April 2010.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

Authors' Addresses

Zach Shelby
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

Brian Frank
SkyFoundry
Richmond, VA
USA

Phone:
Email: brian@skyfoundry.com

Don Sturek
Pacific Gas & Electric
77 Beale Street
San Francisco, CA
USA

Phone: +1-619-504-3615
Email: d.sturek@att.net

CoRE
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2011

Z. Shelby
Sensinode
October 25, 2010

CoRE Link Format
draft-ietf-core-link-format-01

Abstract

This document defines a link format for use by constrained CoAP web servers to describe URIs of resources offered along with other attributes. Based on the HTTP Link Header format, the CoRE link format is carried as a payload and is assigned an Internet media type. A well-known URI is defined as a default entry-point for requesting the list of links to resources hosted by a server.

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 April 28, 2011.

Copyright Notice

Copyright (c) 2010 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	3
2. Link Format	3
2.1. Target and context URIs	4
2.2. Link relation 'rel' usage	5
2.3. Description 'd' usage	5
2.4. Alternative URI 'sh' usage	5
2.5. Resource name 'n' usage	5
2.6. Content-type code 'ct' usage	5
2.7. Resource identifier 'id' usage	6
2.8. Examples	6
3. Well-known Interface	6
3.1. Query Filtering	7
4. Security Considerations	8
5. IANA Considerations	9
5.1. Well-known 'core' URI	9
5.2. New link-format Internet media type	9
6. Acknowledgments	10
7. Changelog	10
8. References	11
8.1. Normative References	11
8.2. Informative References	11
Author's Address	11

1. Introduction

The Constrained RESTful Environments (CoRE) working group 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 [I-D.shelby-core-coap-req].

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 Discovery. In this document we refer to the discovery of resources offered by a CoAP server as resource discovery.

The core function of such a discovery mechanism is to provide URIs ("links") for the resources offered, complemented by information describing the relationship between the resource description and each resource as well as other attributes. When such a collection of attributed resource references (links) is offered as a resource of its own (as opposed to as HTTP headers delivered with a different resource), we speak of its representation as a link-format.

This document specifies a link-format for use in CoRE resource discovery by extending the HTTP Link Header Format [I-D.nottingham-http-link-header] to describe resources hosted by a constrained server. The CoRE link-format is carried as a payload and is assigned an Internet media type. A well-known URI `"/.well-known/core"` is defined as a default entry-point for requesting the list of links to resources hosted by a server.

2. Link Format

CoRE resource discovery extends the HTTP Link Header format specified in [I-D.nottingham-http-link-header] which is specified in Augmented Backus-Naur Form (ABNF) notation [RFC2616]. The format does not require special XML or binary parsing, and is extensible.

This link format is used for a similar purpose to that described in [I-D.nottingham-http-link-header], to describe one or more relationships between resources. However in this specification the link format is extended with specific constrained M2M link parameters, links are carried as a payload rather than in a message header, and a default interface is defined to discover resources described by these links.

[I-D.nottingham-http-link-header] did not require an Internet media type for this link format, as it assumes to be carried in an HTTP header. This specification thus requests a Internet media type for this format (see Section 5.2).

The CoRE link format uses the ABNF description and associated rules in Section 5 of [I-D.nottingham-http-link-header]. In addition, the URI, URI-reference and pchar rules are taken from [RFC3986]. The "Link:" text is omitted as that is part of the HTTP Link Header. Multiple link descriptions are separated by commas. The CoRE link format MUST use the US-ASCII character set (support for RFC2231 encoding of non-ASCII content TBD). The following CoRE specific link-extension parameters to the format are defined:

```
link-extension      = ( "d" "=" <"> URI-reference <"> )
link-extension      = ( "sh" "=" <"> URI-reference <"> )
link-extension      = ( "n" "=" quoted-string )
link-extension      = ( "ct" "=" integer )
link-extension      = ( "id" "=" quoted-string )
integer             = 1*DIGIT
```

2.1. Target and context URIs

Each link description conveys one target URI as a URI-reference inside angle brackets ("<>"). The context URI of a link (also called base URI in [RFC3986]) conveyed in the description is by default the URI of the resource that returned the link-format representation. Thus each link can be thought of as describing a target resource hosted by the server in the absence of further relation information. This is an important difference to the way the HTTP Link Header format is used, as it is included in the header of an HTTP response for some URI (this URI is by default the context). Thus the HTTP Link Header is by default relating the target URI to the URI that was requested. In comparison, the CoRE link format includes one or more link entries, each describing a resource hosted by a server. See Section 5 of [RFC3986] for a description of how URIs are constructed from URI references.

As per Section 5.2 of [I-D.nottingham-http-link-header] a link description MAY include an "anchor" attribute, in which case the context is the URI included in that attribute. This can be used to describe a relationship between two resources. A consuming implementation can however choose to ignore such links. It is not expected that most implementations will be able to derive useful information from explicitly anchored links.

2.2. Link relation 'rel' usage

Link descriptions in CoRE are typically used to describe entry points to services hosted by the server, and thus in the absence of the rel attribute the registered "service" relation type is assumed. In the CoRE link format the service relation type indicates that the link is a service hosted by the server (in the absence of the anchor attribute). A description can make use of any registered relation type or extension types in the form of a URI by including the rel attribute.

2.3. Description 'd' usage

The description "d" attribute can provide a URI to a specific interface definition used to access the target resource. This could be for example a URI to the WADL definition of the target resource. Multiple description attributes MAY appear in a link description.

2.4. Alternative URI 'sh' usage

This attribute can be included to define an alternative short URI which can also be used to access the target resource. Multiple alternative short URI attributes MAY appear in a link description.

2.5. Resource name 'n' usage

The resource name "n" attribute is used to assign either a human readable or a semantically important name to a resource. In the case of a temperature sensor resource the name could be something like "Temperature in Centigrade", a URI to an ontology like "http://sweet.jpl.nasa.gov/2.0/phys.owl#Temperature" or an application-specific semantic name like "TemperatureC". Multiple name attributes MAY appear in a link description.

2.6. Content-type code 'ct' usage

The Content-type code "ct" attribute provides a hint about the Internet media type this resource returns. The value is in the CoAP identifier code format as a decimal ASCII integer [I-D.ietf-core-coap]. For example application/xml would be indicated as "ct=41". If no Content-type code attribute is present then nothing about the type can be assumed. The Content-type code attribute MUST NOT appear more than once in a link description.

Alternatively, the "type" attribute MAY be used to indicate an Internet media type as a quoted-string. It is not however expected that constrained implementations are able to parse quoted-string Content-type values.

2.7. Resource identifier 'id' usage

The resource identifier "id" field is a unique identifier (e.g. UUID or XRI) for this resource for use in e.g. resource or search directories. The resource identifier attribute MUST NOT appear more than once in a link description.

2.8. Examples

A few examples of typical link descriptions in this format follows. Multiple resource descriptions in a representation are separated by commas. Commas can also occur in quoted strings and URIs but do not end a description. Linefeeds never occur in the actual format, but are shown in the example for readability.

This example includes link descriptions for an index to sensors hosted by a server, along with links to two different sensors.

```
GET /.well-known/core
```

```
</sensors>;rel="index";n="Sensor Index",  
</sensors/temp>;sh="/t";n="TemperatureC",  
</sensors/light>;sh="/l";ct=41;n="LightLux"
```

This example arranges link descriptions hierarchically, with the entry point including a link description to a sub-resource containing link descriptions about the sensors.

```
GET /.well-known/core
```

```
</.well-known/core/sensors>;rel="section"  
;type="application/link-format"
```

```
GET /.well-known/core/sensors
```

```
</sensors/temp>;sh="/t";n="TemperatureC",  
</sensors/light>;sh="/l";ct=41;n="LightLux"
```

3. Well-known Interface

Resource discovery in CoRE is accomplished through the use of a well-known resource URI which returns a list of links (resource descriptions) offered by that constrained server. Well-known resources have a path component that begins with "/.well-known/" as specified in [RFC5785]. This document defines a new well-known path prefix for CoRE discovery "/.well-known/core" [Section 5.1]. A

server implementing this specification MUST support this path prefix on the default port appropriate for the protocol for the purpose of resource discovery. It is however up to the application which link descriptions are included and how they are organized. In the absence of any links, a zero-length payload is returned. The resource representation of this resource is described in Section 2.

The CoRE resource discovery interface supports the following interactions:

- o Performing a GET on /.well-known/core to the default port returns a list of link descriptions available from a CoAP server (if any).
- o Filtering may be performed on any of the link format attributes using a query string as specified in Section 3.1. For example [GET /.well-known/core?n=TemperatureC] would request resources with the name TemperatureC. A server is not however required to support filtering.
- o More capable servers such as proxies could support a resource directory by requesting the resource descriptions of other end-points or accepting [POST /.well-known/core messages] from other servers. This adds the resources of other end-points as a sub-resource in which absolute URIs are included for the link-values. The details of such resource directory functionality is however out of scope for this document.

End-points with a large number of resources SHOULD include resource descriptions only for important services or collections and organize their resource descriptions into a hierarchy of link resources. This is done by including links in the /.well-known/core list which point to other resource lists, e.g. </.well-known/core/sensors>. Such a hierarchy SHOULD be under the /.well-known/core path but could be located elsewhere.

3.1. Query Filtering

A server implementing this document MAY recognize the query part of a resource-discovery URI as a filter on the resources to be returned. The query part should conform to the following syntax:

```
filter-query = resource-param "=" query-pattern
resource-param = "uri" | "d" | "sh" | "n" | "id"
query-pattern = 1*pchar [ "*" ]
```

The resource-param "uri" refers to the URI-reference between the "<"

and ">" characters of a link-value. Other resource-param values refer to the link attribute they name. (TBD: Do we want to add the resource description attributes that I excluded, or the standard link-param attributes from I-D.nottingham-http-link-header?) Filtering is performed by comparing the query-pattern against the value of the attribute identified by the resource-param for each link-value in the collection of resources identified by the URI path.

If the decoded query-pattern does not end with "*", a link value matches the query only if the value of the attribute or URI-reference denoted by the resource-param is bitwise identical to the query-pattern. If the decoded query-pattern ends with "*", it is sufficient that the remainder of the query-pattern be a prefix of the value denoted by the resource-param.

It is not expected that very constrained nodes support filtering. Implementations not supporting filtering MUST simply ignore the query string and return the whole resource for unicast requests. An exact match is performed on the query string, and a 200 OK response is returned with link descriptions that contains the matching entries (if any). In contrast, a multicast request with a query string MUST not be responded to if filtering is not supported (to avoid a needless response storm). If resource descriptions are organized hierarchically, a query on the root resource /.well-known/core SHOULD return all matching resource descriptions from the entire hierarchy. An example query on the link descriptions from Section 2 may look like:

```
GET /.well-known/core?n=LightLux
```

```
</sensors/light>;sh="/l";ct=41;n="LightLux"
```

4. Security Considerations

This document needs the same security considerations as described in Section 7 of [I-D.nottingham-http-link-header]. The /.well-known/core resource may be protected e.g. using DTLS when hosted on a CoAP server as per [I-D.ietf-core-coap] Section 10.2.

Great care must be taken when processing multicast requests using CoAP for the well-known link-format resources, as this could be used to perform denial of service on a constrained network. A multicast request SHOULD only be accepted if the request is sufficiently authenticated and secured.

5. IANA Considerations

5.1. Well-known 'core' URI

This memo registers the "core" well-known URI in the Well-Known URI Registry as defined by [RFC5785].

URI suffix: core

Change controller: IETF

Specification document(s): [[this document]]

Related information: None

5.2. New link-format Internet media type

This memo registers the a new Internet media type for the CoRE link format, application/link-format.

Type name: application

Subtype name: link-format

Required parameters: None

Optional parameters: The query string may contain uri= to match the URI, or any other attribute defined for the link format to match that attribute.

Encoding considerations: US-ASCII

Security considerations: None

Interoperability considerations:

Published specification: [[this document]]

Applications that use this media type: CoAP server and client implementations.

Additional information:

Magic number(s):

File extension(s):

Macintosh file type code(s):

Intended usage: COMMON

Restrictions on usage: None

Author: CoRE WG

Change controller: IETF

6. Acknowledgments

Special thanks to Mark Nottingham and Eran Hammer-Lahav for discussions and ideas that led to this draft, and to Carsten Bormann and Peter Bigot for extensive comments and contributions that improved the text.

Thanks to Michael Stuber, Richard Kelsey, Cullen Jennings, Guido Moritz, Peter Van Der Stok, Adriano Pezzuto, Lisa Dussealt, Alexey Melnikov, Gilbert Clark, Salvatore Loreto, Petri Mutka, Szymon Sasin, Robert Quattlebaum, Robert Cragie, Angelo Castellani, Tom Herbst, Ed Beroaset, Gilman Tolle, Robby Simpson, Peter Bigot, Colin O'Flynn and David Ryan for helpful comments and discussions that have shaped the document.

7. Changelog

Changes from ietf-00 to ietf-01:

- o Editorial changes to correct references.
- o Formal definition for filter query string.
- o Removed URI-reference option from "n" and "id".
- o Added security text about multicast requests.

Changes from shelby-00 to ietf-00:

- o Fixed the ABNF link-extension definitions (quotes around URIs, integer definition).
- o Clarified that filtering is optional, and the query string is to be ignored if not supported (and the URL path processed as normally).
- o Required support of wildcard * processing if filtering is supported.

- o Removed the assumption of a default content-type assumption.

8. References

8.1. Normative References

- [I-D.ietf-core-coap]
Shelby, Z., Frank, B., and D. Sturek, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-02 (work in progress), September 2010.
- [I-D.nottingham-http-link-header]
Nottingham, M., "Web Linking", draft-nottingham-http-link-header-10 (work in progress), May 2010.
- [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.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

8.2. Informative References

- [I-D.shelby-core-coap-req]
Shelby, Z., Stuber, M., Sturek, D., Frank, B., and R. Kelsey, "CoAP Requirements and Features", draft-shelby-core-coap-req-02 (work in progress), October 2010.

Author's Address

Zach Shelby
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2011

K. Hartke, Ed.
Universitaet Bremen TZI
Z. Shelby
Sensinode
October 18, 2010

Observing Resources in CoAP
draft-ietf-core-observe-00

Abstract

The state of a resource can change over time. We want to give clients of the CoRE WG CoAP protocol the ability to observe this change. This short I-D provides a design for such an addition to CoAP.

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 April 21, 2011.

Copyright Notice

Copyright (c) 2010 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	3
2. Architecture	4
2.1. Subscriptions	4
2.2. Notifications	4
3. Subscription-lifetime Option	6
3.1. Example	7
4. Open issues	8
5. Acknowledgements	9
6. References	10
6.1. Normative References	10
6.2. Informative References	10
Appendix A. Data types	11
A.1. Variable-length unsigned integer	11
Authors' Addresses	12

1. Introduction

The state of a resource can change over time. We want to give CoAP [I-D.ietf-core-coap] clients the ability to observe this change.

This short I-D describes an architecture and a protocol design that realizes the well-known subject/observer design pattern within the REST-based [REST] environment of CoAP.

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP implementations.

2. Architecture

The architecture is based on the well-known subject/observer design pattern. In this pattern, an object, called the subject, maintains a list of interested parties, called observers, and notifies them automatically when a predefined condition occurs.

In the context of CoAP, the subjects are resources. A subscription to a resource causes the CoAP server to continuously supply an CoAP client with the state of the resource: once upon subscription and then whenever the state of the resource changes.

As with the existing REST methods, this architecture is about exchanging representations of resources, not about the messages (or method calls).

2.1. Subscriptions

A client subscribes to a resource by performing a GET request that includes the Subscription-lifetime Option (Section 3). A subscription request MAY include a Token Option, which will then be included in all subsequent notifications. For robustness, a subscription has to be maintained through periodic refreshing. If a subscription is not refreshed, it MUST end after the duration that is negotiated using the Subscription-lifetime Option. A client refreshes a subscription by repeating the original GET request before the subscription lifetime expired.

2.2. Notifications

Upon subscription, an observer MUST be supplied with the current state of the resource. For efficiency, this initial notification MAY be sent within the same message that acknowledges the subscription request.

The client is notified of resource state changes by additional responses sent from the server to the client. Each such notification response MUST include either the request URI or Token Option, and the remaining subscription lifetime.

It is not necessary that a subscribed client receives every single notification response, or that the server sends a notification response for every single state change. However, the state observed by an observer SHOULD eventually become consistent with the actual state of the observed resource.

The representation format (i.e. the media type) used during the lifetime of a subscription MUST NOT change. If the server is unable

to continue sending notification responses to a client in the requested representation format, it MUST send a response with code 406 (Not Acceptable) and end the subscription.

A server MUST NOT send any further notification responses after sending a response with code 4xx or 5xx, i.e. the subscription MUST end. (Note: a client must be prepared to receive additional notification responses after receiving such a response. In this case, it MUST handle them like a subscription notification that it cannot relate to a subscription.)

For resources that change in a somewhat predictable or regular fashion, it is RECOMMENDED that a notification message is non-confirmable. For robustness, a server MAY instead request the acknowledgment of a notification response from a client by marking it as confirmable. (For example, in order to check if the client is still there, or to make sure that an observer observes a particular resource state.) If a confirmable notification requires a retransmission, it is RECOMMENDED to send the state that is current at the instant of the retransmission; it is NOT RECOMMENDED to have multiple such confirmable notification transactions active for one resource/client pair at any one instant.

If a client cannot relate a confirmable notification response to a subscription, it MUST reject the message with a RST (in which case the server MUST end the subscription). Otherwise, it MUST acknowledge the message with an ACK.

3. Subscription-lifetime Option

Type	C/E	Name	Data type	Length	Default
10	E	Subscription-lifetime	Variable-length unsigned integer (Appendix A.1)	0-4 B	0

The Subscription-lifetime Option, when present, modifies the GET method so it does not only retrieve a representation of the current state of the resource identified by the request URI once, but also lets the server notify the client of changes to the resource state for the duration specified in the option.

(Note: since the Subscription-lifetime Option is elective, the GET request that includes the Subscription-lifetime Option will automatically fall back to a simple GET request if the server does not support subscriptions.)

In a response, the Subscription-lifetime Option indicates a lower bound (e.g., by rounding down) for the remaining subscription lifetime. (Note that the server can always choose to cut short the subscription lifetime before it echoes this lifetime back in an ACK or a confirmable response.)

3.1. Example

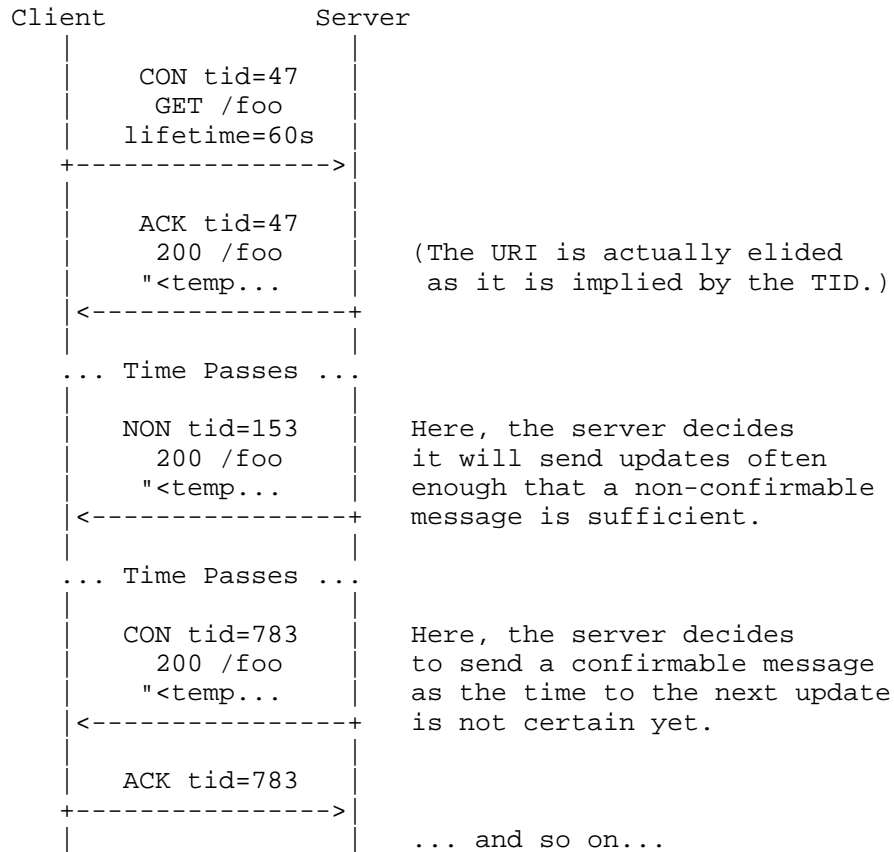


Figure 1

4. Open issues

Add discussion of messages that get reordered.

Add discussion of how to handle the influence of datagram latency on subscription lifetimes.

Describe how subscriptions interact with other CoAP features (e.g., the Block Option, caching, etc.).

Describe how to map subscriptions to HTTP long-polls, WebSockets, and other asynchronous forms of HTTP.

5. Acknowledgements

Carsten Bormann was an original author of this draft and is acknowledged for significant contribution to this document. This work was partially funded by the Klaus Tschira Foundation.

6. References

6.1. Normative References

- [I-D.ietf-core-coap] Shelby, Z., Frank, B., and D. Sturek, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-02 (work in progress), September 2010.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

- [REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>.
- (Seminal dissertation introducing the REST architectural style.)

Appendix A. Data types

A.1. Variable-length unsigned integer

A "Variable-length unsigned integer" is a non-negative integer that is represented in network byte order and uses a variable number of bytes as shown in Figure 2.

```

Len = 0      (implies value of 0)

              0
              0 1 2 3 4 5 6 7
Len = 1      +-----+
              |      0-255      |
              +-----+

              0                               1
              0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
Len = 2      +-----+-----+
              |      0-65535      |
              +-----+-----+

Len = 3 is 24 bits, Len = 4 is 32 bits etc.

```

Figure 2: Variable length unsigned integer value format

Authors' Addresses

Klaus Hartke (editor)
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63905
Fax: +49-421-218-7000
Email: hartke@tzi.org

Zach Shelby
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

Core
Internet-Draft
Intended status: Informational
Expires: May 12, 2011

C. O'Flynn
Atmel Corporation
B. Sarikaya
Huawei USA
Y. Ohba
Toshiba
Z. Cao
China Mobile
R. Cragie
Pacific Gas and Electric
November 8, 2010

Security Bootstrapping of Resource-Constrained Devices
draft-oflynn-core-bootstrapping-03

Abstract

The Internet of Things is marching its way towards completion. Nodes can use standards from the 6LoWPAN and ROLL WG to achieve IP connectivity. IEEE Standards ensure connectivity at lower layers for resource-constrained devices. Yet a central problem remains at a more basic layer without a suitable answer: how to initially configure the network. Without configuration the network never advances beyond a large box of nodes. Current solutions tend to be specific to a certain vendor, node type, or application.

This document outlines exactly what problems are faced in solving this problem. General problems faced in any low-power wireless network are outlined first; followed by how these apply to bootstrapping. A selection of currently proposed techniques is presented. From these a more generic approach is presented, which can solve the problem for a wide range of situations.

An emphasis is on performing this bootstrapping in a secure manner. This document does not cover operation of the network securely. This document does provide the basis for allowing the network to operate securely however, by providing standard methods for key exchanges and authentication.

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 12, 2011.

Copyright Notice

Copyright (c) 2010 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	5
1.1. What is Bootstrapping?	5
1.2. Why IETF?	5
2. Bootstrapping Architecture	6
2.1. Areas of Bootstrapping	6
2.2. Architecture	7
3. Communications Channel	8
3.1. Supported Communication Channels	8
4. Bootstrap Security Method	8
4.1. None	9
4.2. Asymmetric with User Authentication, Followed by Symmetric	9
4.3. Asymmetric with Certificate Authority, Followed by Symmetric	9
4.4. Cryptographically Generated Address Based Address Ownership Verification	9
5. Bootstrap Protocols	9
5.1. System Level Objectives	10
5.2. EAP Authentication Framework	10
5.3. PANA	12
5.4. HIP-DEX	14
5.5. 802.1X	15
6. Security Considerations	16
7. IANA Considerations	16
8. Acknowledgements	16
9. References	16
9.1. Normative References	16
9.2. Informative References	17
Appendix A. Examples of Node Configuration	19
A.1. Smart Energy	19
A.1.1. Initial Meter Installation	19
A.1.2. Home Expansions	19
A.2. Consumer Products	20
A.2.1. Connecting DVD Remote to DVD Player	20
A.2.2. Adding a TV to a network with a DVD player and remote	20
A.2.3. Providing GPS Location Data	20
A.3. Commercial Building Automation	20
A.3.1. Light Installation	20
Appendix B. Example Exchanges	20
B.1. Smart Energy: Meter Manufacture	20
B.2. Smart Energy: Meter Installation	20
B.3. Smart Energy: Home Expansion	21
B.4. Consumer: Connecting DVD Remote to DVD Player	21
B.5. Consumer: Adding a TV to a network with a DVD player and remote	22

B.6. Consumer: Providing GPS Location Data	24
B.7. Commercial: Building Automation	24
Authors' Addresses	24

1. Introduction

Familiarity with constrained network types is assumed here. Documents produced in the 6LoWPAN, ROLL, and CoRE Working Groups (WGs) would be a useful reference for the reader. In particular RFC 4919 [RFC4919] from 6LoWPAN, RFC 5548 [RFC5548] and RFC 5673 [RFC5673] from ROLL, CoAP [I-D.ietf-core-coap] from CoRE, and a paper by Romer and Mattern [ROMER04]. Familiarity with application specific examples such as Zigbee or Smart Energy groups is assumed.

A summary of those will be presented, as far as network requirements are concerned. The general network requirements will be further concentrated into requirements surrounding only the bootstrapping issues.

A number of solutions which are currently in use will be presented. Requirements on each solution will be stated to enable their use as a security bootstrapping protocol.

1.1. What is Bootstrapping?

Node configuration is known as bootstrapping in this document. Bootstrapping is any processing required before the network can operate. Typically this will require a number of settings to be transferred between nodes at all layers. This could include anything from link-layer information (i.e., wireless channels, link-layer encryption keys) to application-layer information (i.e., network names, application encryption keys).

Bootstrapping is complete when settings have been securely transferred prior to normal operation in the network.

1.2. Why IETF?

The bootstrapping problem is not specific to any MAC or PHY. This problem exists across any two nodes which have no previous knowledge of each other. In particular, this problem is complicated when the nodes are resource-constrained and may not have an advanced user interface. The IETF is instrumental in defining standards which will be used by The Internet of Things. Ensuring these standards can be used across nodes and networks requires some form of bootstrapping which any node can use.

Existing standards will be used as much as possible in this document. The method proposed here should work across many different underlying layers. It could be used to allow two nodes on the same physical network to join at the physical layer, or allow two nodes on an incompatible physical network to join at the IPv6 layer.

2. Bootstrapping Architecture

2.1. Areas of Bootstrapping

In order to provide a flexible architecture, the bootstrapping method is split into five distinct areas and two distinct phases. The five areas are a 'user interface', 'bootstrap profile', 'security method', 'bootstrap protocol', and the 'communications channel'.

The phases are provisioning phase and bootstrapping phase. In the provisioning phase, statically configured parameters (e.g., certificates) needed for the bootstrapping phase is provisioned. In the bootstrapping phase, dynamically configured information is set up using the statically configured information provided in the provisioning phase.

The user interface provides both user input and user output. Simple nodes may only have a push-button and LED, more complex nodes may have a graphical display and keyboard. The user interface provides interaction between the user and bootstrapping methods. The user interface would be used during bootstrapping as an OOB channel. It may also be used to specify bootstrapping policies.

The user interface provides the interaction between the user and the bootstrap protocol. The user interface will vary depending on the capabilities of the node. Examples might include a push-button and LED on simple nodes, to full-blown graphical user interfaces. Note that a 'bootstrapping tool' used to initially deploy a network is just a special user interface. This allows a very uniform protocol in deployment and use of networks.

User interface is out-of-scope and will not be further discussed.

Two nodes communicate through some channel. For our purposes this is split into the 'control channel' and 'data channel'. The control channel is used for the bootstrap protocol, and the data channel is used during normal network operation. A node may support multiple control or data channels. When the control and data channels are the same, the bootstrapping is done In Band (IB). When the control and data channels are different, the bootstrapping is performed Out Of Band (OOB). An 802.15.4 network for instance would use an 802.15.4 control channel for IB bootstrapping, but a control channel of perhaps IrDA or USB for OOB bootstrapping.

The 'bootstrap profile', i.e. statically configured parameters during the provisioning phase, defines what information should be exchanged during the process. A single node may run the protocol multiple times with different profiles. If the user wishes to associate a new

lightswitch, the protocol is first run with the '802.15.4 Wireless Profile', through which it learns the channel and PAN-ID. The node then runs a 'Security Exchange Profile' to learn the needed encryption keys. Finally it runs a 'Lightswitch Association Profile' through which it learns which light to associate with.

The 'security method' defines supported security methods for bootstrapping. The supported security methods will depend on the control channel and bootstrap profile. In one node if the control channel is secure, then a simple clear-text security method is supported. For example when a physical connection between two nodes is used, the control channel is considered secure. However when the control channel is not secure, this clear-text security method is not supported. The 'bootstrap profile' additionally defines allowed security methods. Higher security nodes may outlaw ever performing a clear-text exchange, even if the control channel is deemed secure.

The 'bootstrap protocol' defines the actual messages exchanged during bootstrapping. The messages are used to transfer between nodes data, node information, and network state. The selected security method runs on top of the control channel, such as EAP-GPSK etc.

2.2. Architecture

Security bootstrapping architecture is structured in a hierarchy of nodes going from the least resource constraint to the most resource constraint. At the top there is a root node. The root node is called Coordinator or Trust Center in Zigbee and 6LowWAN Border Router (6LBR) in 6LoWPAN ND.

At the next level there are interior Routers. Routers are able to run a routing protocol between other routers and the root. Router are called 6LowWAN Routers (6BR) in 6LoWPAN ND.

At the lowest level there are the nodes. The nodes do not run a routing protocol. They can connect to the nearest router over a single radio link. The nodes are called End Device in Zigbee and host in in 6LoWPAN ND.

Routers first join the network as a node and go through security bootstrapping operations in order to create a Master Session Key (MSK). Next routers execute routing protocol, e.g. [I-D.ietf-roll-rpl] specific steps to create session keys with their neighbors and to establish upstream and downstream next hop parents.

At each node hierarchy level described above, there are lower-layer and higher-layer protocols to bootstrap their ciphering keys, where the lower-layer refers to layers below IP layer including IEEE

802.15.4 MAC layer and LoWPAN adaptation layer and the higher-layer refers to IP layer and the above. In general, required bootstrapping procedures depend on the bootstrapping protocols to use. Section 5 describes the bootstrapping procedures where EAP (Extensible Authentication Protocol) [RFC3748] and other protocols are used as the bootstrapping protocols.

3. Communications Channel

The communications channel is the method used between two nodes to communicate. There are two main communication channels: the 'control' and 'data' channels. The control channel is used during bootstrapping, and the data channel is used during network operation.

3.1. Supported Communication Channels

There is no limit on what communications channels are supported. The following gives an example of several supported channels:

- o IEEE 802.15.4
- o Power-Line Communications
- o IrDA
- o RFID
- o Some simple physical link
- o Cellular
- o Ethernet
- o IPv6
- o Wi-Fi

Depending on the node's function, it may use different channels as the data or control channel. Nodes may have multiple data and/or control channels as well.

4. Bootstrap Security Method

The bootstrap security method defines allowable security methods. A node may choose to support or use a subset of these methods. This is NOT the security architecture used for the application, but only the

security used during bootstrapping. Typically some high-security method is used to generate a shared secret, which then switches to simpler symmetric encryption to secure the actual bootstrapping channel. The techniques negotiated should take advantage of hardware resources available, such as hardware encryption accelerators on an end node.

4.1. None

This is the simplest security method. No encryption or authentication is provided, messages are exchanged completely in clear-text. It is assumed some other layer provides security, such as a physical connection between devices.

4.2. Asymmetric with User Authentication, Followed by Symmetric

A Diffie-Hellman style key exchange is used to generate a shared secret. The authentication will be provided by the user, by confirming cryptographic signatures between two devices. With the shared secret generated through the DH, some symmetric encryption is used to secure the actual bootstrapping channel.

4.3. Asymmetric with Certificate Authority, Followed by Symmetric

Public key exchanges are used (aka: DH again), but with a Certificate Authority. Once a shared secret exists, symmetric encryption is used to secure the actual bootstrapping channel.

4.4. Cryptographically Generated Address Based Address Ownership Verification

A node may generate the global unique address using different techniques other than the stateless address autoconfiguration. For example, Cryptographically Generated Addresses (CGA) [RFC3972] is a type of global unique address that can be used to verify the address ownership. When the node uses CGA, it MUST execute SeND protocol [RFC3971]. In a 6LOWPAN network, a modified 6LOWPAN ND Protocol [I-D.ietf-6lowpan-nd] must be executed between the node and the edge router.

5. Bootstrap Protocols

In this section we first present system level objectives that security bootstrapping protocols are expected to achieve. Next, we present EAP authentication framework and then describe three different protocols.

5.1. System Level Objectives

Authentication/ reauthentication: nodes joining the network MUST at the first place authenticate to the trust center. In order to achieve secure multi-hop routing, the node MUST authenticate to its upstream and downstream neighbors. A bootstrapping solution MUST support re-authentication of resource-constrained devices and re-keying of dynamically generated keys.

Data Confidentiality: the data communication between two endpoints MAY be encrypted using the derived key, avoiding being eavesdropped by a non-trusted third part.

Data Integrity: the data communication between two endpoints MUST NOT be altered by some intermediate nodes. The nodes should be able to detect the non-integral data.

Keys and key freshness: the keys used for data communication MUST have a lifetime, in order to keep their freshness. A bootstrapping solution MUST support both symmetric and asymmetric key authentication. If distribution of a key to be used for a resource-constrained device is required, a bootstrapping solution MUST support secure key distribution to prevent the key from eavesdropping, alternation and replay attacks.

Multi domain support: A bootstrapping solution MUST be able to allow resource-constrained devices that may be subscribed to different administrative domains to be connected to the same access network at the same time.

Identities: A bootstrapping solution MUST be able to allow a resource-constrained device to use various types of identities used for authentication, including device identities, user identities or combinations of different types of identities. Also a bootstrapping solution MUST be able to allow a resource-constrained device to change its identities used for authentication over time.

Authentication infrastructure: A bootstrapping solution MUST be able to operate with or without an authentication infrastructure.

5.2. EAP Authentication Framework

In EAP, there are three distinct entities: the client or EAP peer, the authenticator and the authentication or EAP server [RFC5247].

The EAP peer is the node that requires to be authenticated before being admitted to the network. The authentication server is the device authenticating the node for bootstrapping. The authenticator

is the device that is admitting the node to the network and it resides in between the peer and authentication server.

EAP client and EAP server exchange EAP messages to execute the authentication algorithm, a.k.a. EAP method. The authenticator is responsible for forwarding EAP messages between the client and server. In 802.1X, EAP messages are carried in Layer 2 and in PANA in IP or Layer 3. EAP messages between the authenticator and authentication server are carried using AAA protocols (RADIUS or Diameter).

At the end of a successful EAP method execution a master session key (MSK) is generated at both the EAP peer and EAP server. Authenticator receives MSK from EAP server at the end of EAP method execution using key transport. MSK is used in deriving a session key between the node and the authenticator using a protocol called secure association protocol (SAP). Derivation of the session key terminates bootstrapping of a node.

Additional keying material derived between EAP client and server that is exported by the EAP method is called Extended Master Session Key (EMSK). EMSK is not used in session key derivation but it could be used for the needs of other applications in higher layer protocols.

In the architecture introduced in Section 2.2 the node or router is the peer and the root is the authenticator. When the supplicant and authenticator are one hop away the authenticator can be reached directly. However, this is rarely the case. In other cases the authenticator authenticates neighboring supplicants first. The router nodes that are authenticated become relay authenticators in the next phase and they relay authentication messages from the supplicants to the authenticator and vice versa. This continues until all nodes are authenticated.

EAP is a lock-step protocol, i.e. it executes in pairs of EAP-Request messages sent by the server and EAP-Response messages sent by the peer. At the end, the server indicates the status of authentication, usually by EAP-Success message which also carries the MSK. The first EAP-Request/Response pair is used for the server to request the identity and the peer to provide it. In the other pairs of EAP exchanges EAP method is executed.

Several EAP methods have been standardized each for different purposes. To authenticate devices with certificates, EAP Transport Layer Security (TLS) v1.2 specified in [RFC5216] which supports certificate-based mutual authentication is used.

Smart Energy Profile 2.0 Application Protocol Specification [SE2.0]

mandates each device to be factory programmed with a certificate. The certificate is bound to a unique network ID, e.g. the device's MAC address or EUI-64 address. During EAP-Identity exchange the EAP peer provides its EUI-64 address as an identity to EAP server. This enables the server to validate the device certificate.

5.3. PANA

PANA (Protocol for carrying Authentication for Network Access) [RFC5191] defines an EAP transport over UDP where a PANA Client (PaC) is an EAP peer and a PANA Authentication Agent (PAA) is an EAP authenticator. There are three bootstrapping scenarios using PANA.

1. Use of PANA for bootstrapping link-layer security.

In this case, PANA is used for network access authentication to bootstrap link-layer ciphering. Security for higher-layer (i.e., IP layer and above) protocols is bootstrapped from an IB or OOB mechanism other than PANA. For example, in a 6LoWPAN deployment PANA authentication can take place to bootstrap IEEE 802.15.4 MAC layer ciphering keys. In ZigBee IP, IEEE 802.15.4 MAC layer ciphering keys used as session keys are derived from a group key so called a Network Key that is securely distributed to each joining node upon successful PANA authentication using AES key wrap over PANA [I-D.ohba-pana-keywrap] where the key encryption key is derived from the EAP MSK (Master Session Key) [RFC3748].

2. Use of PANA for bootstrapping higher-layer security.

In this scenario, PANA is used as an OOB mechanism for higher-layer authentication to bootstrap ciphering keys for one or more higher-layer protocols independently of network access authentication. The PAA may reside in a higher-layer network element such as an ANSI C12.22 authentication host [C1222] and a CoAP server, or an independent server dedicated for service authentication for multiple higher-layer protocols. When bootstrapping ANSI C12.22 security for which no IB key management mechanism is available, ANSI C12.22 ciphering keys are directly derived from EAP key material established from PANA authentication. When bootstrapping CoAP security with DTLS protection, a PSK (Pre-Shared Key) credential in the combined usage of DTLS (Datagram Transport Layer Security) [RFC4347] and PSK mode of TLS [RFC4279] is derived from EAP key material and DTLS ciphering keys are generated as a result of a successful DTLS handshake. Similarly, when bootstrapping CoAP security with IPsec ESP protection, a PSK credential of IKEv2 [RFC5996] is derived from EAP key material and IPsec ESP ciphering keys are generated as a result of a successful IKEv2 handshake.

The ability to bootstrap multiple higher-layer protocols from a single execution of PANA authentication is important to save the computational resources for resource-constrained devices especially where public-key based authentication is used.

3. Use of PANA for bootstrapping both link-layer and higher-layer security.

This case is the combination of the other two cases, and the most optimized way for bootstrapping resource-constrained devices. This case is only applicable where both the network access authentication and the higher-layer authentication use the same authentication server with the same authentication credentials.

The second and third scenarios are generally referred to as Single Sign-On in section 4.2.2.2 of [NISTIR7628VOL1], where the root keys for higher-layer protocols can be derived from EAP EMSK (Extended Master Session Key) as an USRK (Usage-Specific Root Key) [RFC5295].

A PANA Relay Element (PRE) is needed to enable PANA messaging between a PANA Client (PaC) which is the node to be authenticated and a PANA Authentication Agent (PAA) which is the authenticator where the two nodes cannot reach each other by means of regular IP routing. This happens during authentication since only a link-local IPv6 address can be used prior to the completion of a successful authentication.

PRE which is one hop away from PaC receives PANA messages and relays the message contents (payload) by encapsulating it in a message parameter called Attribute Value Pair (AVP). PRE also needs to send header contents such as PaC's IP address and UDP port number in a different AVP. PRE has IP routing established with PAA which could be several hops away. PAA sends its reply messages in which the payload is encapsulated in an AVP. It also adds the AVP containing PaC's IP address and UDP port number. PRE sends creates a link local PDU using these AVPs and sends it to PaC.

The requirements for the use of PANA as a bootstrapping protocol can be stated as follows:

- o A new entity called PANA Relay Element needs to be added to the PANA architecture. Behaviour of PANA Relay Element needs to be defined.
- o New AVPs needed for PANA Relay Element operation for properly relaying messages from the client to the authenticator and vice versa are required to be specified.

- o An extension to PANA to securely distribute keys from the PANA Authentication Agent to the PANA Client using AES Key Wrap with Padding algorithm needs to be defined. This is needed in order to use PANA for group key distribution.

5.4. HIP-DEX

[RFC4423] introduces the Host Identity Protocol (HIP) where the Host Identity (HI) is a Cryptographic key (RSA, DSA, or ECC). A 128-bit length Host Identity Tag (HIT) is derived from the HI (hashed) and functions as an IPv6 address (/128 prefix) for applications. A four-packet Peer-to-Peer Host Identity Protocol Base EXchange (HIP BEX) establishes a security association (SA, similar to IKE), indexed by the HITs, but independent of the IP address. So HIP can be considered as a shim layer between the transport(TCP/UDP) and IP, providing authentication, data confidentiality, mobility in one basket.

The HIP-BEX involves many crypto primitives that are difficult to run on constrained nodes. HIP Diet Exchange (HIP-DEX) [I-D.moskowitz-hip-rg-dex] is a way to make HIP lightweight. Basically, HIP-DEX a variant of the HIP-BEX specifically designed to use as few crypto primitives as possible yet still provide the same class of security features as HIP-BEX.

HIP-DEX can be used for mutual authentication between two endpoints. After mutual authentication, the two endpoints establish a shared secret, which is fresh and fed into the encryption algorithm for data confidentiality. So HIP-DEX can achieve the authentication, key freshness and data confidentiality objectives of security bootstrapping.

When a node wants to authenticate to the network using HIP and Diet-HIP, it should be able to complete the HIP-BEX or HIP-DEX with the trust anchor or some delegate. In HIP, it does not matter how many domains, and nodes can authenticate each other as long as they have the secret.

In the architecture introduced in Section 2.2 the node and router could be the HIP end-points. Depending on who initiates the HIP Diet Exchange, the node or router could act as the HIP initiator and HIP responder respectively. And the initiator and responder can be multiple hops way from each other, as long as there is an IP connectivity between them.

An important requirement for the HIP-DEX to work in the architecture, the initiator should be able to get the IP address of the responder, either using DNS infrastructure or local configuration.

5.5. 802.1X

IEEE 802.1X defines how EAP packets can be transported over in Layer 2, i.e. Ethernet frames [802.1x] by encapsulating EAP packets into EAP Over Lan (EAPOL) frames between EAP peer, called supplicant and the authenticator. EAPOL can also be used in 802.11 wireless links.

To enable IEEE 802.15.4 devices to use EAP authentication, EAP packets encapsulated in EAPOL frames can be carried as payload in 802.15.4 data frames [802.15.4]. EAPOL is well defined and widely used and it lends itself to be easily carried in 802.15.4 data frames. For this, Frame Type subfield of the Frame Control Field of IEEE 802.15.4 MAC header needs to be set to a special value to indicate the type of the payload, i.e. 802.1X encapsulated EAP packets. EAPOL packets are encoded following common EAPOL PDU structure defined in [802.1x] into the data payload field of 802.15.4 data frames.

Authentication proceeds as follows: authenticator authenticates the supplicants that are on the next hop first. This enables a secure link between the authenticator and these first-hop nodes. First-hop nodes or router become Relay Authenticators in the next phase of authentication. Relay authenticators tunnel EAPOL frames to the authenticator in the secure link established. This way all the supplicants are gradually authenticated.

The keys established from a successful EAP method (such as PSK mode of TLS), the node runs neighbor discovery protocol to get an IPv6 address assigned [I-D.ietf-6lowpan-nd]. Data transfer can be secured using DTLS or IPSec. Keys derived from EAP TLS are used in either generating DTLS ciphering keys after a successful DTLS handshake or IPSec ESP ciphering keys after a successful IKEv2 handshake.

802.1X can achieve the authentication, key freshness and data confidentiality objectives of security bootstrapping. Multi domain operation is intrinsically supported due to the use of EAP and AAA.

The requirements for the use of 802.1X defined EAPOL as a bootstrapping protocol can be stated as follows:

- o A special value in the Frame Type subfield of the Frame Control Field of IEEE 802.15.4 MAC header to indicate the type of the payload,
- o Group addresses for 802.15.4 corresponding to EAPOL Group Address Assignments defined in Table 11.1 of [802.1x], especially to be used in EAPOL-Start packet.

- o Which MAC frames of beacon, data, acknowledgment and MAC command as defined in [802.15.4] with what security levels are mapped to controlled port, which MAC frames with what security levels are mapped to uncontrolled port and which MAC frames are never mapped to any of controlled/uncontrolled port (i.e., the payload of those frames are used by the MAC-layer itself and never used by upper layers).

6. Security Considerations

TBD.

7. IANA Considerations

This memo includes no request to IANA.

8. Acknowledgements

Thanks to Zach Shelby for editing, comments, and overall assistance. Special thanks also to Rene Struik and Carsten Borman for their comments that helped us improve the writing.

9. References

9.1. Normative References

- [802.15.4] IEEE Std 802.15.4-2006, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)", September 2006.
- [802.1x] IEEE Std 802.1X-2010, "IEEE 802.1X Port-Based Network Access Control", February 2010.
- [RF4CE] ZigBee Alliance, "Zigbee RF4CE Specification Version 1.00", March 2009.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.
- [RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschafenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.
- [RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.
- [RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.
- [RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.
- [ROMER04] Romer, K. and F. Mattern, "The design space of wireless sensor networks", IEEE Wireless Communications, vol. 11, no. 6, pp. 54-61, December 2004.
- [SE2.0] ZigBee Alliance, "Smart Energy Profile 2.0 Technical Requirements Document", April 2010.

9.2. Informative References

- [C1222] American National Standard, "Protocol Specification For Interfacing to Data Communication Networks", ANSI C12.22-2008, 2008.
- [I-D.ietf-6lowpan-nd]
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low-power and Lossy Networks", draft-ietf-6lowpan-nd-14 (work in progress), October 2010.
- [I-D.ietf-core-coap]
Shelby, Z., Frank, B., and D. Sturek, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-03 (work in progress), October 2010.
- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., Brandt, A., Clausen, T., Hui, J., Kelsey, R., Levis, P., Networks, D., Struik, R., and J. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-ietf-roll-rpl-14 (work in progress), October 2010.

progress), October 2010.

[I-D.moskowitz-hip-rg-dex]

Moskowitz, R., "HIP Diet EXchange (DEX)",
draft-moskowitz-hip-rg-dex-02 (work in progress),
July 2010.

[I-D.narten-iana-considerations-rfc2434bis]

Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs",
draft-narten-iana-considerations-rfc2434bis-09 (work in
progress), March 2008.

[I-D.ohba-pana-keywrap]

Chakrabarti, S., Cragie, R., Duffy, P., Ohba, Y., and A.
Yegin, "Protocol for Carrying Authentication for Network
Access (PANA) Extension for Key Wrap",
draft-ohba-pana-keywrap-01 (work in progress),
October 2010.

[NISTIR7628VOL1]

The Smart Grid Interoperability Panel - Cyber Security
Working Group, "Guidelines for Smart Grid Cyber Security:
Vol. 1, Smart Grid Cyber Security Strategy, Architecture,
and High-Level Requirements", NISTIR 7628, vol. 1, 2010.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC
Text on Security Considerations", BCP 72, RFC 3552,
July 2003.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure
Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)",
RFC 3972, March 2005.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites
for Transport Layer Security (TLS)", RFC 4279,
December 2005.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
Security", RFC 4347, April 2006.

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol
(HIP) Architecture", RFC 4423, May 2006.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.
- [RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, August 2008.
- [RFC5433] Clancy, T. and H. Tschofenig, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", RFC 5433, February 2009.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

Appendix A. Examples of Node Configuration

Before any detail on methods is explored, the following section will provide various examples this document could cover. Exact requirements will be brought forward in subsequent sections. For the reader's general understanding this section is placed to give an idea of an acceptable usage scenario.

A.1. Smart Energy

A.1.1. Initial Meter Installation

The meter is initially loaded with code and network keys through a physical interface at the factory. The meter is installed at a customers home, and configured by the installer through the backbone link (via GSM modem, etc). Both operations can be performed through methods defined herein.

A.1.2. Home Expansions

The user wishes to join a thermostat onto the network. They press a button on the thermostat, which enters join mode. They press a button on the smart meter, which allows nodes to join the network. The devices both have displays, so they display a certain number which the user verifies match on both devices. The thermostat has now securely joined the network.

A.2. Consumer Products

A.2.1. Connecting DVD Remote to DVD Player

The user pushes a join button on the DVD remote and DVD player. The devices find each other, and blink in unison to indicate to the user which two devices will join. The user presses the button to confirm this, and the two devices are now joined together.

A.2.2. Adding a TV to a network with a DVD player and remote

The user then presses the join button on the DVD player and TV. The devices again find each other and blink in unison, with the addition that the remote control also blinks to indicate it is present in the network.

A.2.3. Providing GPS Location Data

A user has a simple GPS receiver (that has no user interface) they wish to broadcast location data with. The user switches on their camera, and enters a PIN from the base of the GPS. The user can now view GPS information such as satellite health from their camera. In addition photos are automatically tagged with location information.

A.3. Commercial Building Automation

A.3.1. Light Installation

The electrician installs the light fixture. Each light has a barcode printed on it. They use a handheld barcode scanner tool, which acts as the commissioning tool. When they scan a barcode with the tool, the tool asks the electrician to enter some additional information such as light fixture location. The tool securely registers the light fixture on the network, along with setting parameters inside the light fixture.

Appendix B. Example Exchanges

The following details how the protocol handles certain conditions and situations through examples. Note that each example is a more detailed description of the examples in Appendix A.

B.1. Smart Energy: Meter Manufacture

B.2. Smart Energy: Meter Installation

B.3. Smart Energy: Home Expansion

B.4. Consumer: Connecting DVD Remote to DVD Player

Supported User Interface Profiles

Profile	DVD Player	Remote Control
none	Y	Y
simple	Y	Y
numerical	Y	N
alphanumeric	Y	N
Graphical	Y	N

Supported Bootstrap Transport Layers

Layer	DVD Player	Remote Control
Physical	Y	Y
802.15.4	Y	Y
IrDA	Y	N

Supported Security Methods

Method	DVD Player	Remote Control
None	Y	Y
EAP	Y	N
Asymmetric, User	Y	Y
Asymmetric, CA	Y	N

The DVD player and remote control have a number of ways in which they could be joined together. The remote control does not have any unique identifier printed on it, thus no pre-shared key can be identified. This leaves either an unsecure joining method, or some asymmetric security method.

The remote control has a button on it for 'join', as does the DVD player. The user pushes the button on the DVD player, and then pushes the button on the remote control. Based on the UI profile, this causes the following to occur:

- o DVD Player scans for existing network in advertise mode. Finding none, it starts a new network and that network enters advertise mode.
- o The DVD remote scans for a network, and then finds the DVD player's network.
- o The devices generate a shared secret (ie: Diffie-Hellman), and both blink their LED in a unique pattern based on this shared secret.
- o The user confirms both devices are blinking the same pattern, as both LEDs are blinking in unison.
- o The DVD player displays 'JOIN OK' on it's LCD panel.

B.5. Consumer: Adding a TV to a network with a DVD player and remote

This network will have three devices: a TV, a DVD Player, and a Remote Control. The user will run the bootstrap protocol between the TV and Remote Control in this example, although it could also be run between the TV and DVD player.

Supported User Interface Profiles

Profile	TV	Remote Control
none	Y	Y
simple	Y	Y
numerical	Y	N
alphanumeric	Y	N
Graphical	Y	N

Supported Bootstrap Transport Layers

Layer	TV	Remote Control
Physical	Y	Y
802.15.4	Y	Y
IrDA	Y	N

Supported Security Methods

Method	TV	Remote Control
None	Y	Y
EAP-GPSK	Y	N
Asymmetric, User	Y	Y
Asymmetric, CA	Y	N

The TV and remote control have a number of ways in which they could be joined together. The remote control does not have any unique identifier printed on it, thus no pre-shared key can be identified. This leaves either an unsecure joining method, or some asymmetric security method.

The remote control has a button on it for 'join', as does the TV. In this example two sequence will be considered: where the TV button is pressed first, and where the remote control button is pressed first.

If the TV join button is pressed first:

- o TV scans for existing networks in advertise mode. Finding none, it starts a new network and that network enters advertise mode.
- o The remote scans for a network, and then finds the TV's network.
- o The remote informs the TV it is on an existing network, and thus will require the TV to join this network.
- o The devices generate a shared secret, and both blink their LED in a unique pattern.
- o The DVD player in addition blinks, so the user is informed that if they confirm the join action the resulting network will have all three devices in it.
- o The user confirms both devices are blinking the same pattern, as both LEDs are blinking in unison.
- o The TV displays 'JOIN OK' onscreen, along with any information about the network it just joined.

If the remote control join button is pressed first:

- o Remote control scans for existing networks in advertise mode. Finding none, it advertises it's network.

- o The TV scans for a network, and then finds the remote control's network.
- o The devices generate a shared secret, and both blink their LED in a unique pattern.
- o The DVD player in addition blinks, so the user is informed that if they confirm the join action the resulting network will have all three devices in it.
- o The user confirms both devices are blinking the same pattern, as both LEDs are blinking in unison.
- o The TV displays 'JOIN OK' onscreen, along with any information about the network it just joined.

B.6. Consumer: Providing GPS Location Data

B.7. Commercial: Building Automation

Authors' Addresses

Colin Patrick O'Flynn
Atmel Corporation
Colorado Springs, Colorado
USA

Phone:
Email: colin.oflynn@atmel.com

Behcet Sarikaya
Huawei USA
1700 Alma Dr. Suite 500
Plano, TX 75075

Email: sarikaya@ieee.org

Yoshihiro Ohba
Toshiba
Tokyo, Japan

Email: yoshihiro.ohba@toshiba.co.jp

Zhen Cao
China Mobile
Beijing, China

Email: caozhen@chinamobile.com

Robert Cragie
Pacific Gas and Electric
89 Greenfield Crescent
Wakefield, UK WF4 4WA

Email: robert.cragie@gridmerge.com

CoRE
Internet-Draft
Intended status: Informational
Expires: April 14, 2012

A. Rahman, Ed.
InterDigital Communications, LLC
E. Dijk, Ed.
Philips Research
October 12, 2011

Group Communication for CoAP
draft-rahman-core-groupcomm-07

Abstract

This is a working document intended to trigger discussion and develop draft text for the CoAP protocol specification in the area of group communication. Engineering tradeoffs become more challenging in constrained environments, therefore group communication is considered within the context of adjacent topics that may impact or be impacted by design choices in the subject area. A solution based on IP multicast is proposed.

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 April 14, 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. Conventions and Terminology	3
2. Introduction	3
2.1. Background	3
2.2. Problem Statement and Scope	4
3. Potential Solutions	5
3.1. Overview	5
3.2. Requirements	6
3.3. IP Multicast	8
3.3.1. Multicast Listener Discovery (MLD) and Multicast Router Discovery (MRD)	8
3.3.2. Group URIs and Multicast Addresses	9
3.3.3. Group Discovery	9
3.3.4. Group Resource Manipulation	10
3.3.5. IP Multicast Transmission Methods	11
3.3.6. Congestion Control	12
3.4. Overlay Multicast	13
3.5. CoAP Application Layer Group Management	13
3.6. CoAP Multicast and HTTP Unicast Interworking	16
3.7. CoAP-Observe for Group Communication	18
4. Recommended Solution	18
4.1. Overview	19
4.2. An Example Protocol Flow	19
4.3. Implementation in Target Network Topologies	22
4.3.1. Single LLN Topology	23
4.3.2. Single LLN with Backbone Topology	25
4.3.3. Multiple LLNs with Backbone Topology	27
4.3.4. LLN(s) with Multiple 6LBRs	27
4.3.5. Conclusions	27
4.4. HTTP/CoAP Interworking Aspects	28
4.5. Implementation Considerations	28
4.5.1. MLD Implementation on LLNs	28
4.5.2. 6LBR Implementation	29
4.5.3. Backbone IP Multicast Infrastructure	29
5. Security Considerations	30
6. IANA Considerations	31
7. Conclusions	31
8. Acknowledgements	31
9. References	32
9.1. Normative References	32
9.2. Informative References	33
Authors' Addresses	35

1. Conventions and 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 [RFC2119].

The following are definitions of specific terminology used in this draft.

Group Communication: A source node sends a message to more than one destination node, where all destinations are identified to belong to a specific group. The set of source nodes and destination nodes may consist of an arbitrary mix of constrained and non-constrained nodes. Sending methods may include serial unicast, multicast, or hybrid unicast-to-multicast solutions.

Multicast: Sending a message to multiple receiving nodes simultaneously. Typically, this is done as part of a group communication process. There are various options to implement multicast including layer 2 (Media Access Control) or layer 3 (IP) mechanisms.

IP Multicast: A specific multicast solution based on the use of IP multicast addresses as defined in "IANA Guidelines for IPv4 Multicast Address Assignments" [RFC5771] and "IP Version 6 Addressing Architecture" [RFC4291].

Low power and Lossy Network (LLN): LLNs are made up of constrained devices. These devices may be interconnected by a variety of links, such as IEEE 802.15.4, Bluetooth, WiFi, wired or low-power powerline communication links.

2. Introduction

2.1. Background

The CoRE working group is chartered to design and standardize a Constrained Application Protocol (CoAP) for resource constrained devices and networks [I-D.ietf-core-coap]. The requirements for CoAP are documented in [I-D.shelby-core-coap-req].

Constrained devices can be large in number, but highly correlated to each other. For example, all the light switches in a building may belong to one group and all the thermostats belong to another group. All the smart meters in the same region can belong to a group as well. Groups may be composed by function; for example, the group "all lights in building one" may consist of the groups "all lights on

floor one of building one", "all lights on floor two of building one", etc. Groups may also be configured or dynamically formed.

In this draft, we focus and expand discussions on the requirements pertaining to CoAP "group communication" and "multicast" support including:

REQ 9: CoAP will support a non-reliable IP multicast message to be sent to a group of Devices to manipulate a resource on all the Devices simultaneously. The use of multicast to query and advertise descriptions must be supported, along with the support of unicast responses.

Currently, the CoAP protocol [I-D.ietf-core-coap] supports unreliable IP multicast using UDP. It defines the unreliable multicast operation as follows:

"CoAP supports sending messages to multicast destination addresses. Such multicast messages MUST be Non-Confirmable. Mechanisms for avoiding congestion from multicast requests are being considered in [I-D.eggert-core-congestion-control]."

Additional requirements were introduced in [I-D.vanderstok-core-bc] driven by quality of experience issues in commercial lighting; the need for large numbers of devices to respond with near simultaneity to a command (multicast PUT), and for that command to be received reliably (reliable multicast).

2.2. Problem Statement and Scope

In this draft, we expand the scope from unreliable IP multicast in the current CoAP requirement to group communication, using either (reliable or unreliable) multicast or unicast or combinations thereof. We assume that all, or a substantial part of, devices participating in group communication are constrained devices (e.g. such as Low Power and Lossy Network (LLN) devices).

Machine-to-Machine (M2M) networks may contain groups of nodes that are highly correlated (e.g. by type or location). For example, all smart meters in a region may belong to one group, and all light switches in a building control system belong to another. Group communication mechanisms can improve efficiency and latency of communication and reduce bandwidth requirements for a given application.

In the following sections, we address the issues related to group communication in detail, with requirements, proposed solutions and analysis of their impact to the CoAP protocol and implementations.

3. Potential Solutions

3.1. Overview

The classic concept of group communications is that of a single source distributing content to multiple recipients that are all part of a group, as shown in the example sequence diagram in Figure 1. Also shown there is the pre-requisite step of forming the group before content can be distributed to it.

Group communication solutions have evolved from "bottom" to "top", i.e., from the network layer (IP multicast) to application layer group communication, also referred to as application layer multicast. A study published in 2005 [STUDY1] identified new solutions in the "middle" (referred to as overlay multicast) that utilize an infrastructure based on proxies.

Each of these classes of solutions may be compared [STUDY1] using metrics such as link stress and level of host complexity [STUDY2]. The results show for a realistic internet topology that IP Multicast is most resource-efficient, with the only downside being that it requires most effort to deploy in the infrastructure.

The approach adopted in this section is to begin with group communication requirements. This is followed by the solutions of IP multicast, an overlay multicast solution, and application layer group communication. Finally additional topics are covered such as group management and CoAP/HTTP proxies in group communication.

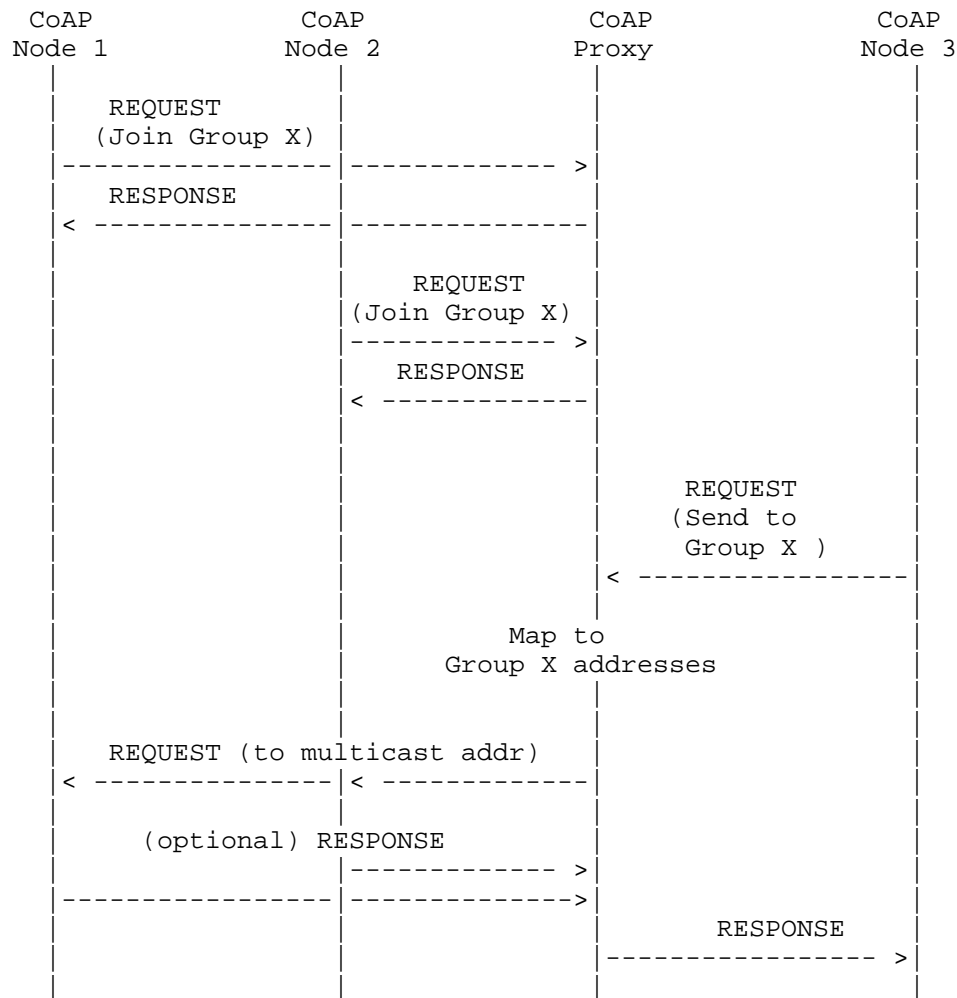


Figure 1: CoAP Group Communications

3.2. Requirements

Requirements that a group communication solution in CoRE should fulfill can be found in existing documents [RFC 5867] [draft-ietf-6lowpan-routing-requirements] [I-D.vanderstok-core-bc] [I-D.shelby-core-coap-req]. Below, a set of high-level requirements is listed that a group communication solution in CoRE should ideally fulfill. More precise requirements may depend on the chosen application (area).

A CoRE group communication solution should (ideally) offer:

- REQ 1: Optional Reliability: unreliable group communication, with preferably reliable group communication as an option.
- REQ 2: Efficiency: delivers messages more efficiently than a "serial unicast only" solution. Also, it should provide a right balance between group data traffic and control overhead.
- REQ 3: Low latency: deliver a message (preferably) as fast as possible.
- REQ 4: Synchrony: allows near-simultaneous modification of a resource on all devices in a group, providing to users a perceived effect of synchrony or simultaneity. It can be expressed as a time span D such that message m is delivered to all destinations in a time interval $[t, t+D]$ for arbitrary t .
- REQ 5: Ordering: message ordering in the reliable group communication mode.
- REQ 6: Security: see Section 5 for security requirements for group communication.
- REQ 7: Flexibility: support for one or many source(s), for dense and sparse networks, for high or low listener density, one or many group(s), and multi-group membership.
- REQ 8: Robust group management: includes functionality to join groups, leave groups, view group membership, and persistent group membership in failing node or sleeping node situations.
- REQ 9: Network layer independence: a solution should be specified independent from specific unicast and/or IP multicast routing protocols. It should support different routing protocols and implementations thereof.
- REQ 10: Minimal specification overhead: a group communication solution should preferably re-use existing/established (IETF) protocols that are suitable for LLN deployments, instead of defining new protocols from scratch.
- REQ 11: Minimal implementation overhead: e.g. a solution allows to re-use existing (software) components that are already present on constrained nodes such as (typical) 6LoWPAN/CoAP nodes.
- REQ 12: Mixed backbone/LLN topology support: a solution should work within a single LLN, and in combined LLN/backbone network topologies, including multi-LLN topologies. Both the senders and receivers of CoAP group messages may be attached to different network links or be part of different LLNs, possibly with routers or switches in between group members. In addition, different routing protocols may operate on the LLN and backbone networks. Preferably a solution also works with existing, common backbone IP infrastructure (e.g. switches or routers).

- REQ 13: CoAP Proxying support: a CoAP proxy can handle distribution of a message to a group on behalf of a (constrained) CoAP client.
- REQ 14: Suitable for operation on LLNs with constrained nodes.

3.3. IP Multicast

IP Multicast protocols have been evolving for decades, resulting in proposed standards such as Protocol Independent Multicast - Sparse Mode (PIM-SM) [RFC4601]. Yet, due to various technical and marketing reasons, IP Multicast is not widely deployed on the general Internet. However, IP Multicast is popular in specific deployments such as in enterprise networks (e.g. for video conferencing or general IP multicast PC applications within a single LAN broadcast domain) and carrier IPTV deployments. Therefore, the packet economy and minimal host complexity of IP multicast make it worth investigating for group communication in constrained environments.

3.3.1. Multicast Listener Discovery (MLD) and Multicast Router Discovery (MRD)

The Multicast Listener Discovery (MLD) protocol [RFC3810] (or its IPv4 pendant IGMP) is today the method of choice used by an (IP multicast enabled) IPv6 router to discover the presence of multicast listeners on directly attached links, and to discover which multicast addresses are of interest to those listening nodes. It was specifically designed to cope with fairly dynamic situations in which multicast listeners may join and leave at any time.

IGMP/MLD Snooping is a technique implemented in some corporate LAN routing/switching devices. An MLD snooping switch listens to MLD State Change Report messages from MLD listeners on attached links. Based on this, the switch learns on what LAN segments there is interest for what IP multicast traffic. If the switch receives at some point a multicast packet, it uses the stored information to decide onto which LAN segment(s) to send the packet. This improves network efficiency compared to the regular switch behavior of forwarding every incoming multicast packet onto all LAN segments. An MLD snooping switch may also send out MLD Query messages (which is normally done by an MLD Router) if no MLD router is present.

The Multicast Router Discovery (MRD) protocol [RFC4286] defines a way to discover multicast routers, for the purpose of using this information by IGMP/MLD snooping devices. However, it appears that this protocol is not as commonly implemented in existing products as MLD is.

3.3.2. Group URIs and Multicast Addresses

An approach to map group authorities onto IP multicast addresses using DNS was proposed in [I-D.vanderstok-core-bc]. Examples of group URI naming (and scoping) for a building control application are shown below. Group URIs MUST follow the approach of the URI structure defined in [RFC3986].

URI authority	Targeted group
all.bldg6...	"all nodes in building 6"
all.west.bldg6...	"all nodes in west wing, building 6"
all.floor1.west.bldg6...	"all nodes in floor 1, west wing, etc."
all.bu036.floor1.west.bldg6...	"all nodes in office bu036, floor1, etc."

The authority portion of the URI is used to identify a node (or group) and the resulting DNS name is bound to a unicast or multicast IP address. Each example group URI shown above can be mapped to a unique multicast IP address. This may be an address allocated according to [RFC3956], [RFC3306] or [RFC3307].

3.3.3. Group Discovery

CoAP defines a resource discovery capability but, in the absence of a standardized group communication infrastructure, it is limited to link-local scope IP multicast; examples may be found in [I-D.ietf-core-link-format]. A service discovery capability is required to extend discovery to other subnets and scale beyond a certain point, as originally proposed in [I-D.vanderstok-core-bc].

DNS-based Service Discovery [I-D.cheshire-dnsext-dns-sd] defines a conventional way to configure DNS PTR, SRV, and TXT records to enable enumeration of services, such as services offered by CoAP nodes, or enumeration of all CoAP nodes, within specified subdomains. A service is specified by a name of the form <Instance>.<ServiceType>.<Domain>, where the service type for CoAP nodes is `_coap._udp` and the domain is a DNS domain name that identifies a group as in the examples above. For each CoAP end-point in a group, a PTR record with the name `_coap._udp` or alternatively the name `_coap._udp.<Domain>` is defined and it points to an SRV record having the `<Instance>.<ServiceType>.<Domain>` name.

All CoAP nodes in a given subdomain may be enumerated by sending a query for PTR records named `_coap._udp` to the authoritative DNS server for that zone. A list of SRV records is returned. Each SRV record contains the port and host name (AAAA record) of a CoAP node. The IP address of the node 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. This can be used to store information about the device, e.g. schema=DALI, type=switch, group=lighting.bldg6, etc.

Another feature of DNS-SD is the ability to specify service subtypes using PTR records. For example, one could represent all the CoAP groups in a subdomain by PTR records with the name `_group._sub._coap._udp` or alternatively `_group._sub._coap._udp.<Domain>`.

3.3.4. Group Resource Manipulation

At least two forms of group resource manipulation must be supported. The first is push (multicast PUT or MPUT for short) as e.g. "turn off all the lights simultaneously". Logically, this is similar to publishing a value to multiple subscribers. The second operation is pull (multicast GET or MGET), which is essential for discovery during commissioning and can be illustrated by the example "return all the resources on all CoAP servers advertized by their .well-known/core URI". MGET to an "all-nodes" or "all-CoAP-nodes" multicast IP address should perhaps be limited in scope to link-local multicast for scaling [TBD: and possibly for security reasons, e.g. DoS attacks].

Conceptually, the result of a multicast GET or PUT should be the same as if the client had unicast them serially (that is, a set of {URI, representation} tuples). Practically, there are major benefits to avoiding serial unicast in favor of a multicast CoAP GET/PUT solution:

- packet economy on constrained networks
- M2M resource discovery (solves the "chicken-and-egg" problem)
- apparent simultaneity of events (e.g. in lighting applications)
- average lower latency per event (e.g. in lighting applications)

Ideally, all nodes in a given group (defined by its multicast IP address) must receive the same request with high probability. This will not be the case if there is diversity in the authority port (i.e. a diversity of dynamic port addresses across the group) or if the targeted resource is located at different paths on different nodes. Extending the definition of group membership to include port and path discovery is not desirable.

Therefore, some measures must be present to ensure uniformity in port number and resource name/location within a group.

A first solution in this respect is to couple groups to service

descriptions in DNS (using DNS-SD as in Section 3.3.3 and [I-D.vanderstok-core-bc]). A service description for a multicast group may have a TXT record in DNS defining a schema X (e.g. "schema=DALI"), which defines by service standard X (e.g. "DALI") which resources a node supporting X MUST have. Therefore a multicast source can safely refer to all resources with corresponding operations as prescribed by standard X. For port numbers (which can be found using DNS-SD also) the same holds. Alternatively, only the default CoAP port may be used in all requests.

A second solution is to impose the following restrictions, e.g. for groups not found using, or advertized in, DNS-SD:

- o All CoAP multicast requests MUST be sent to the well-known CoAP port.
- o All CoAP multicast requests SHOULD operate on /.well-known/core URIs

One question is whether the application (or middleboxes) need to be aware that a request is intended for a group. A separate scheme as proposed by [ID.goland-http-udp] might be useful (e.g. "corem" vs. "core"). To the extent that group membership might be implemented as a series of IP multicast, serial unicast, or some combination, having a distinct scheme for group operations might be a useful signal for a proxy receiving the request to look up the group membership and replicate serial unicasts as well as send multicast packets.

3.3.5. IP Multicast Transmission Methods

3.3.5.1. Serial unicast

Even in systems that generally support IP Multicast, there may be certain data links (or transports) that don't support IP multicast. For those links a serial unicast alternative must be provided. This implies that it should be possible to enumerate the members of a group, in order to determine the correct unicast destinations.

3.3.5.2. Unreliable IP Multicast

The CoRE WG charter specified support for non-reliable IP multicast. In the current CoAP protocol design [I-D.ietf-core-coap], unreliable multicast is realized by the source sending Non-Confirmable messages to a multicast IP address. IP Multicast (using UDP) in itself is unreliable, unless specific reliability features are added to it.

3.3.5.3. Reliable IP Multicast

[TBD: This is a difficult problem. Need to investigate the benefits of repeating MGET and MPUT requests (saturation) to get "Pretty Good

Reliability". Use the same MID or a new MID for repeated requests? Carsten suggests the use of bloom filters to suppress duplicate responses.

One could argue that non-idempotent operations (POST) cannot be supported without a *truly* reliable multicast protocol. However, is this the case? If a multicast POST request is sent repeatedly with the same Message ID (MID), then CoAP nodes that already received it once will ignore duplicates. Sending with Message ID is supported in CoAP for Non-Confirmable messages (thus including multicast messages) as per [I-D.ietf-core-coap] section 4.2.]

Reliable multicast supports guaranteed delivery of messages to a group of nodes. The following specifies the requirements as was proposed originally in version 01 of [I-D.vanderstok-core-bc]:

- o Validity - If sender sends a message, *m*, to a group, *g*, of destinations, a path exists between sender and destinations, and the sender and destinations are correct, all destinations in *g* eventually receive *m*.
- o Integrity - destination receives *m* at most once from sender and only if sender sent *m* to a group including destination.
- o Agreement - If a correct destination of *g* receives *m*, then all correct destinations of *g* receive *m*.
- o Timeliness - For real-time control of devices, there is a known constant *D* such that if *m* is sent at time *t*, no correct destination receives *m* after *t*+*D*.

There are various approaches to achieve reliability, such as

- o Destination node sends response: a destination sends a CoAP Response upon multicast Request reception (it SHOULD be a Non-Confirmable response). The source node may retry a request to destination nodes that did not respond in time with a CoAP response.
- o Route redundancy
- o Source node transmits multiple times (destinations do not respond)

3.3.6. Congestion Control

CoAP requests may be multicast, resulting a multitude of replies from different nodes, potentially causing congestion. [I-D.eggert-core-congestion-control] suggests to conservatively control sending multicast requests.

CoAP already addresses the congestion problem to some extent by requiring all multicast CoAP requests to be Non-Confirmable. However, as responses to multicast requests (both MGET or MPUT) are required in CoAP, using CoAP multicast still may lead to congestion issues.

Various means can be implemented to prevent congestion.

[TBD: if an MGET or MPUT request leads to the sending of a CoAP response by servers, the servers should enforce a random delay within TIMEOUT before sending their responses. More investigation required.]

Currently in the CoAP protocol, a MAX_RETRANSMIT value set by default to 4 is used for retransmission of Confirmable messages. Since CoAP multicast messages are Non-Confirmable, no retransmissions will occur in CoAP, making the effective retransmission value 0.

3.4. Overlay Multicast

An alternative group communication solution (to IP Multicast) is an "overlay multicast" approach. We define an overlay multicast as one that utilizes an infrastructure based on proxies (rather than an IP router based IP multicast backbone) to deliver IP multicast packets to end devices. MLD (Section 3.3.1) has been selected as the basis for multicast support by the ROLL working group for the RPL routing protocol. Therefore, it is proposed that "IGMP/MLD Proxying" [RFC4605] be used as a basis for an overlay multicast solution for CoAP.

Specifically, a CoAP proxy [I-D.ietf-core-coap] may also contain an MLD Proxy function. All CoAP devices that want to join a given IP multicast group would then send an MLD Join to the CoAP (MLD) proxy. Thereafter, the CoAP (MLD) proxy would be responsible for delivering any IP multicast message to the subscribed CoAP devices. This will require modifications to the existing [RFC4605] functionality.

Note that the CoAP (MLD) proxy may or may not be connected to an external IP multicast enabled backbone. The key function for the CoAP (MLD) proxy is to distribute CoAP generated multicast packets even in the absence of router support for multicast.

3.5. CoAP Application Layer Group Management

Another alternative solution (to IP Multicast and Overlay Multicast) is to define CoAP application level group management primitives. Thus, CoAP can support group management features without need for any underlying IP multicast support.

Interestingly, such group management primitives could also be offered even if there is underlying IP multicast support. This is useful because IP multicast inherently does not support the concept of a group with managed members, while a managed group may be required for some applications.

The following group management primitives are in general useful:

- o discover groups;
- o query group properties (e.g. related resource descriptions);
- o create a group;
- o remove a group;
- o add a group member;
- o remove a group member;
- o enumerate group members;
- o security and access control primitives.

In this proposal a (at least one) CoAP Proxy node is responsible for group membership management. A constrained node can specify which group it intends to join (or leave) using a CoAP request to the appropriate CoAP Proxy. To Join, the group name will be included in optional request header fields (explained below). These header fields will be included in a PUT request to the Proxy. The Proxy-URI is set to the Group Management URI of the Proxy (found previously through the "/.well-known/" resource discovery mechanism). Note that in this solution also CoAP Proxies may exist in a network that are not capable of CoAP group operations.

Group names may be defined as arbitrary strings with a predefined maximum length (e.g. 268 characters or the maximum string length in a CoAP Option), or as URIs.

[TBD: how can a client send a request to a group? Does it only need to know the group name (string or URI) or also an IP multicast address? One way is to send a CoAP request to the CoAP Proxy with a group URI directly in the Proxy-URI field. This avoids having to know anything related to IP multicast addresses.]

This solution in principle supports both unreliable and reliable group communication. A client would indicate unreliable communication by sending a CoAP Non-Confirmable request to the CoAP Proxy, or reliable communication by sending a CoAP Confirmable request.

It is proposed that CoAP supports two Header Options for group "Join" and "Leave". These Options are Elective so they should be assigned an even number. Assuming the Type for "join" is x (value TBD), the Header Options are illustrated by the table in Figure 2:

Type	C/E	Name	Data type	Length	Default
x	E	Group Join	String	1-270 B	" "
x+2	E	Group Leave	String	1-270 B	" "

Figure 2: CoAP Header Options for Group Management

Figure 3 illustrates how a node can join or leave a group using the Header Options in a CoAP message:

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								
+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+									
Ver T OC										Code										Message ID																			
+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+									
delta length										Join Group A (ID or URI)																													
+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+									
0 length										Join Group B (ID or URI)																													
+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+									
2 length										Leave Group C (ID or URI)																													
+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+										+--+--+--+--+--+--+--+--+--+									

Figure 3: CoAP Message for Group Management

Header Fields for the above example:

Ver: 2-bit unsigned integer for CoAP Version. Set to 1 by implementation as defined by the CoAP specification.

T: 2-bit unsigned integer for CoAP Transaction Type. Either '0' Confirmation or '1' Non-Confirmable can be used for group "join" or "leave" request.

OC: 4-bit unsigned integer for Option Count. For this example, the value should be "3" since there are three option fields.

Code: 8-bit unsigned integer to indicate the Method in a Request or a Response Code in a Response message. Any Code can be used so the

group management can be piggy-backed in either Request or Response message.

Message ID: 16-bit value assigned by the source to uniquely identify a pair of Request and Response.

CoAP defines a delta encoding for header options. The first delta is the "Type" for group join in this specific example. If the type for group join is x as illustrated in Figure 3, delta will be x. In the second header option, it is also a group join so the delta is 0. The third header option is a group leave so the delta is 2.

An alternative solution to using Header Options (explained above) is to use designated parameters in the query part of the URI in the Proxy-URI field of a POST (TBD: or PUT?) request to a Proxy's group management service resource advertized by DNS-SD. For example, to join group1 and leave group2:

```
coap://proxyl.bld2.example.com/groupmgt?j=group1&l=group2
```

3.6. CoAP Multicast and HTTP Unicast Interworking

Within the constrained network, CoAP runs over UDP for which IP multicast is supported. In a non-constrained network (i.e. general Internet), HTTP over TCP is used for which IP multicast is not supported. Therefore a CoAP/HTTP Proxy node that supports group communication needs to have functionalities to support interworking of unicast and multicast. One possible way of operation of the Proxy is illustrated in Figure 4. Note that this topic is covered in more detail in [I-D.castellani-core-http-mapping].

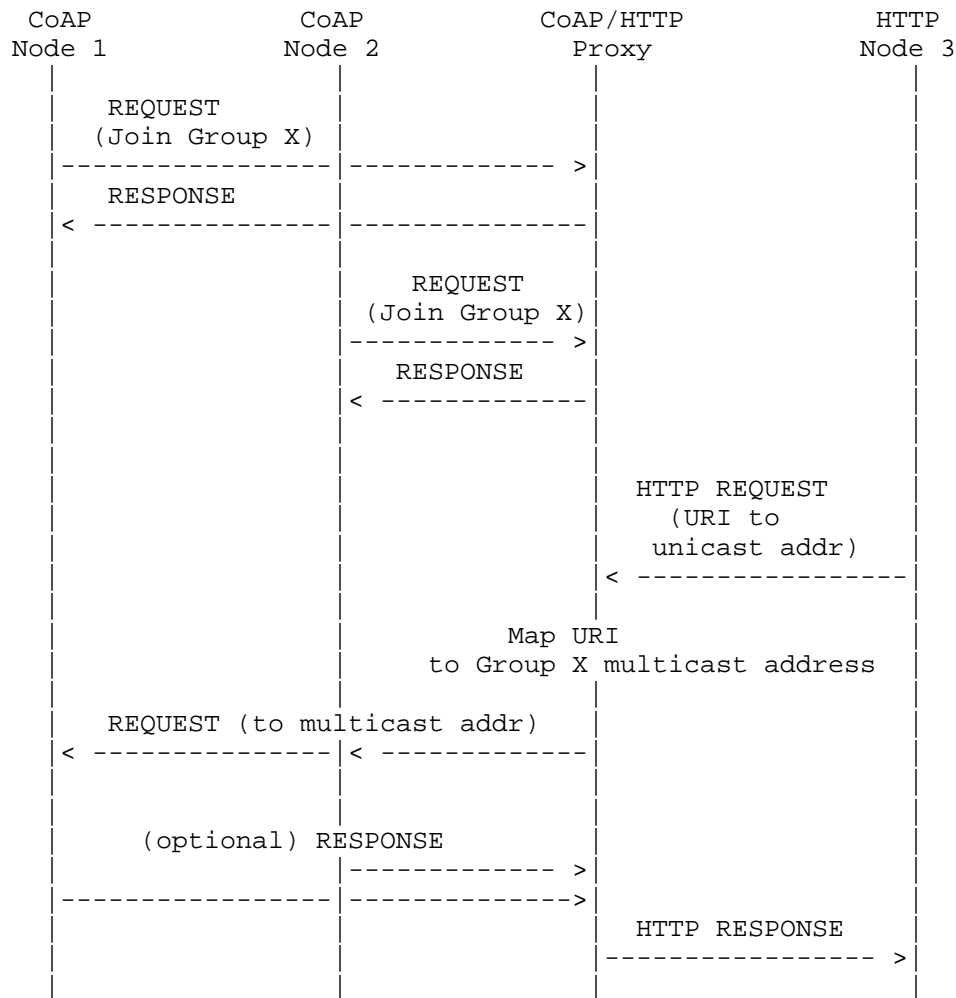


Figure 4: CoAP Multicast and HTTP Unicast Interworking

Note that Figure 4 illustrates the case of IP multicast as the underlying group communications mechanism. However the overlay multicast group communication (Section 3.4) or CoAP application group communication (Section 3.5) can be used as the underlying mechanism and the principles of the figure would still apply (i.e. CoAP proxy needs to do interworking between HTTP unicast and CoAP multicast).

A key point in Figure 4 is that the incoming HTTP Request (from node 3) will carry a URI (with the HTTP scheme) that resolves in the general Internet to the proxy node. At the proxy node, the URI will

then possibly be mapped (as detailed in [I-D.castellani-core-http-mapping]) and again resolved (with the CoAP scheme) to an IP multicast destination. This may be accomplished, for example, by using DNS-SD (Section 3.3.3). The proxy node will then IP multicast the CoAP Request (corresponding to the received HTTP Request) to the appropriate nodes (i.e. nodes 1 and 2).

In terms of the HTTP Response, Figure 4 illustrates that it will be generated by the proxy node based on aggregated responses of the CoAP nodes and sent back to the client in the general Internet that sent the HTTP Request (i.e. node 1). In [I-D.castellani-core-http-mapping] the HTTP Response that the Proxy may use to aggregate multiple CoAP responses is described in more detail. So in terms of overall operation, the CoAP proxy can be considered to be a "non-transparent" proxy according to [RFC2616]. Specifically, [RFC2616] states that a "non-transparent proxy is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction or anonymity filtering."

An alternative to the above is using a Forward Proxy. In this case, the CoAP request URI could be carried in the HTTP Request Line (as defined in [I-D.ietf-core-coap] Section 8) in a HTTP request sent to the IP address of the Proxy.

3.7. CoAP-Observe for Group Communication

The CoAP Observation extension [I-D.ietf-core-observe] can be directly used for group communication. A group then consists of a CoAP server hosting a specific resource, plus all CoAP clients observing that resource. The server is the only group member that can send a group message. It does this by modifying the state of a resource under observation and subsequently notifying its observers of the change. Serial unicast is used in this case for notifications.

Group communication is unreliable in the sense that, even though confirmable CoAP messages may be used, there are no guarantees that an update will be received. For example, a client may believe it is observing a resource while in reality the server rebooted and lost its listener state.

4. Recommended Solution

4.1. Overview

We recommend that IP multicast as outlined in Section 3.3 be adopted as the base solution for CoAP Group Communication. This approach re-uses the IP multicast suite of protocols and can operate on both constrained and non-constrained network segments. The group communication can hence work regardless of the underlying networking technology. Still, this approach may require specifying or implementing additional IP Multicast functionality in an LLN, in a backbone network, or in both - this will be evaluated in more detail in this section.

4.2. An Example Protocol Flow

We first present an example use case to illustrate the overall steps in an IP Multicast based CoAP Group Communication solution. We assume the following network configuration for this example (see Figure 5):

- 1) A large room (Room-A) with three lights (Light-1, Light-2, Light-3) controlled by a Light Switch. The devices are organized into two 6LoWPAN subnets.
- 2) Light-1 and the Light Switch are connected to a router (Rtr-1) which is also a CoAP Proxy and a 6LoWPAN Border Router (6LBR).
- 3) Light-2 and the Light-3 are connected to another router (Rtr-2) which is also a CoAP Proxy and a 6LBR.
- 4) The routers are connected to a an IPv6 network backbone which is also multicast enabled. In the general case, this means the network backbone and 6LBRs support a PIM based multicast routing protocol, and MLD for forming groups. In a limited case, if the network backbone is one link, then the routers only have to support MLD-snooping for the example use case to work.

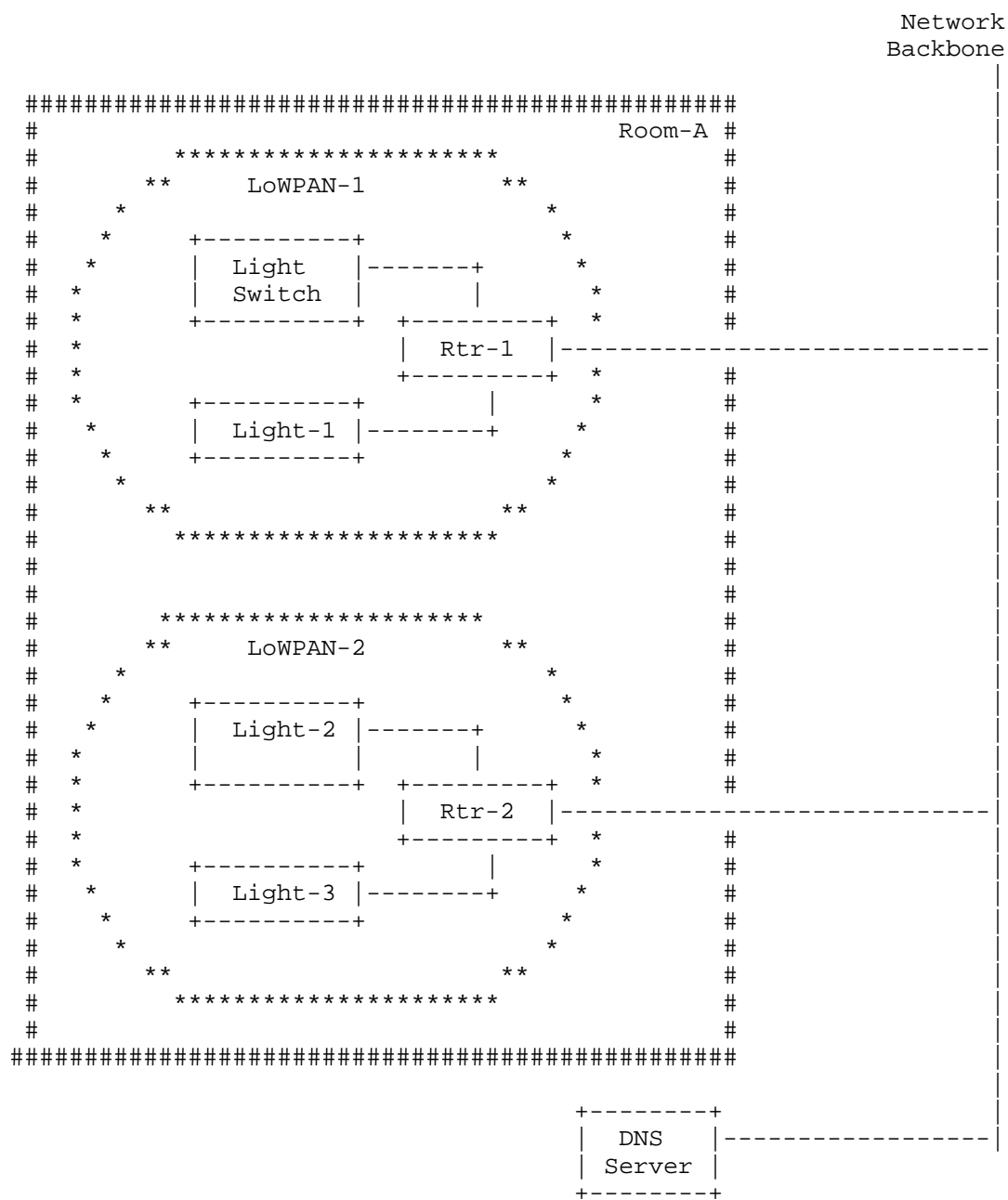
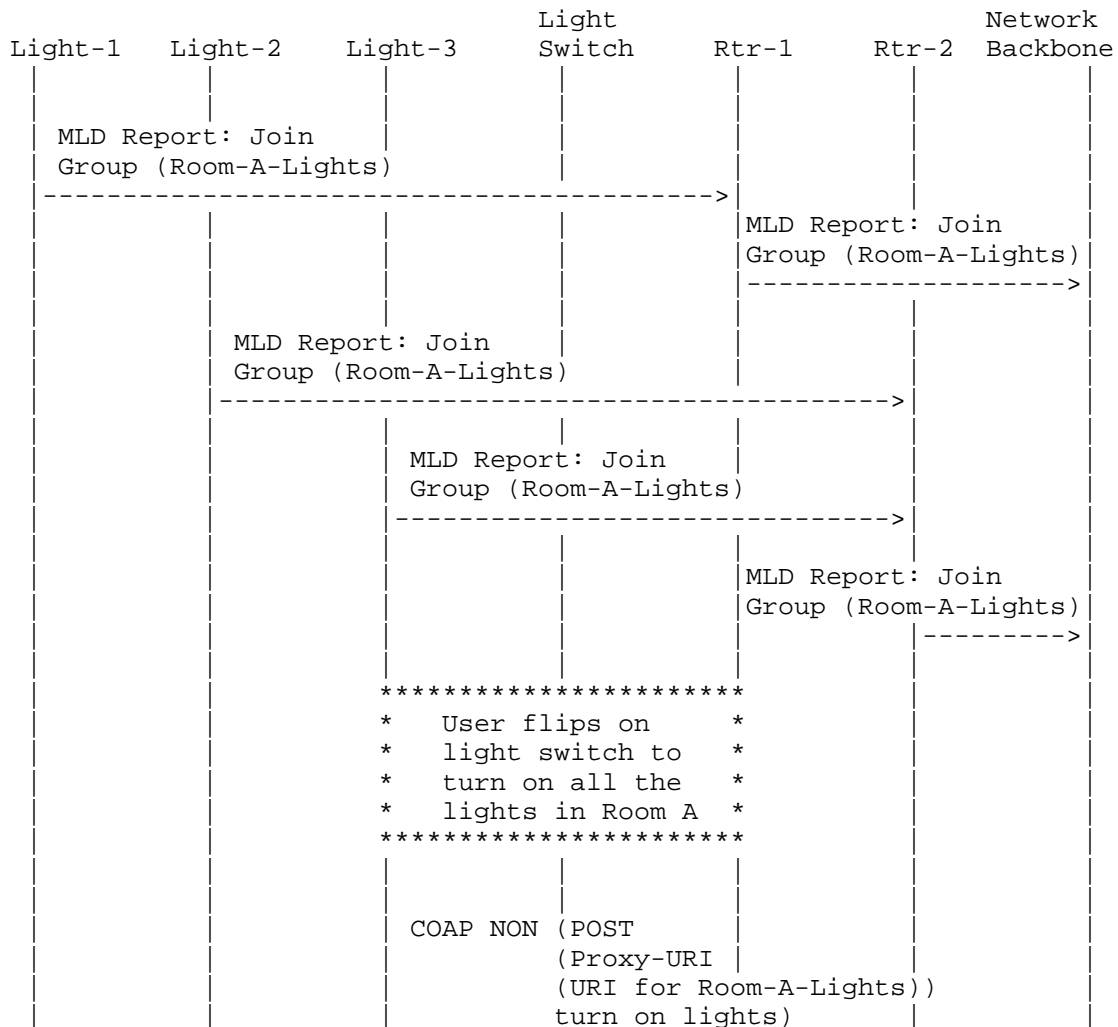


Figure 5: Network Topology of a Large Room (Room-A)

The corresponding protocol flow for an IP Multicast based CoAP Group Communication solution for the network shown in Figure 5 is shown in Figure 6. We assume the following steps occur before the illustrated flow:

1) Startup phase: 6LoWPANs are formed. IPv6 addresses assigned to all devices. The CoAP network is formed.

2) Commissioning phase (by applications): The IP multicast address of the group (Room-A-Lights) has been set in all the Lights. The URI of the group (Room-A-Lights) has been set in the Light Switch.



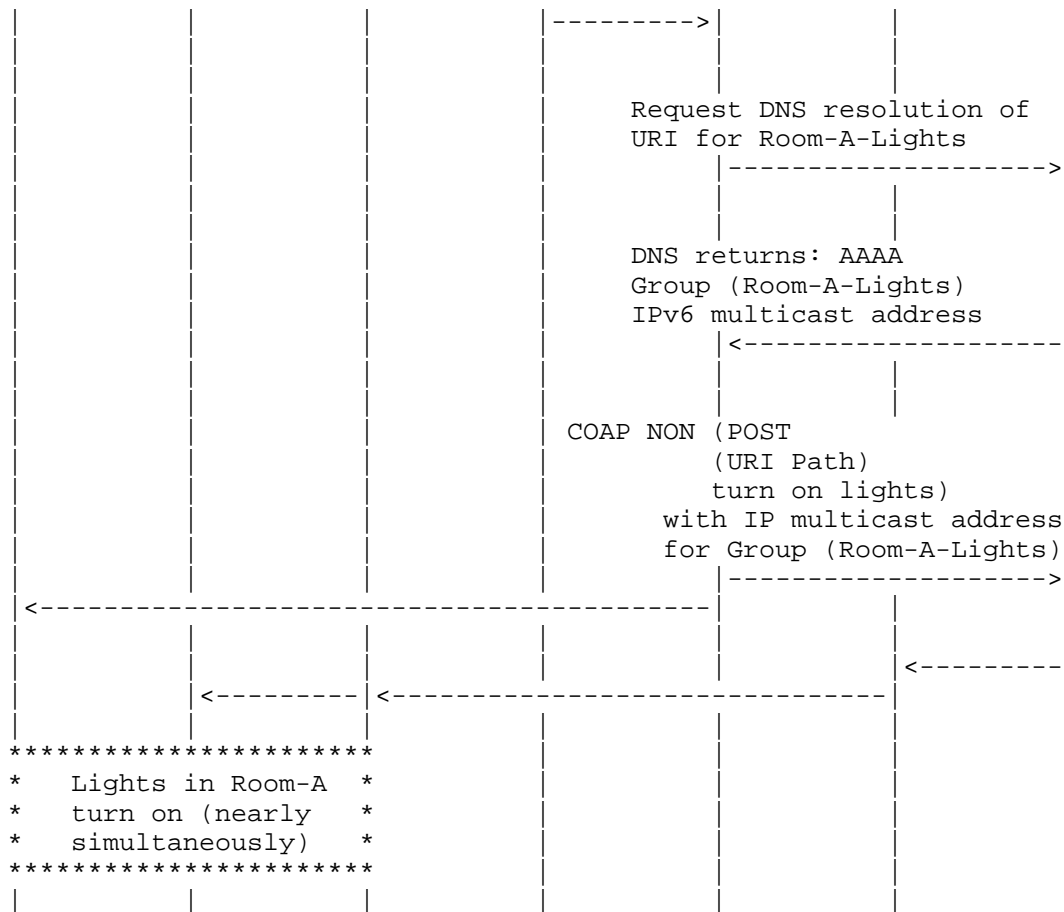


Figure 6: Turning on Lights in a Large Room (Room-A)

The indicated MLD Report messages are link-local multicast. In each LoWPAN, it is assumed that a multicast routing protocol in 6LRs will propagate the Join information over multiple hops to the 6LBR.

4.3. Implementation in Target Network Topologies

This section looks in more detail how an IP Multicast based solution can be deployed onto the various network topologies that we consider important for group communication use cases. Note that the chosen solution of IP Multicast for CoAP group communication works mostly independently from the underlying network topology and its specific IP multicast implementation.

Starting from the simplest case of a single LLN topology, we move to more complex topologies involving a backbone network or multiple LLNs. With "backbone" we refer here typically to a corporate LAN or VLAN, which constitutes a single broadcast domain by design. It could also be an in-home network. A multi-link backbone is also possible, if there is proper IP multicast routing or forwarding configured between these links. (The term 6LoWPAN Border Router or "6LBR" is used here for a border router, though our evaluation is not necessarily restricted to 6LoWPAN networks.)

4.3.1. Single LLN Topology

The simplest topology is a single LLN, where all the IP multicast source(s) and destinations are constrained nodes within this same LLN. Possible implementations of IP multicast routing and group administration for this topology are listed below.

4.3.1.1. Mesh-Under Multicast Routing

The LLN may be set up in either a mesh-under or a route-over configuration. In the former case, the mesh routing protocol should take care of routing IP multicast messages throughout the LLN.

Because conceptually all nodes in the LLN are attached to a single link, there is in principle no need for nodes to announce their interest in multicast IP addresses via MLD (see Section 3.3.1). A multicast message to a specific IP destination, which is delivered to all 6LoWPAN nodes by the mesh routing algorithm, is accepted by the IP network layer of that node only if it is listening on that specific multicast IP address and port.

4.3.1.2. RPL Multicast Routing

The RPL routing protocol for LLNs provides support for routing to multicast IP destinations (Section 12 of [I-D.ietf-roll-rpl]). Like regular unicast destinations, multicast destinations are advertised by nodes using RPL DAO messages. This functionality requires "Storing mode with multicast support" (Mode Of Operation, MOP is 3) in the RPL network.

Once all RPL routing tables in the network are populated, any RPL node can send packets to an IP multicast destination. The RPL protocol performs distribution of multicast packet both upward towards the DODAG root and downwards into the DODAG.

The text in Section 12 of the RPL specification clearly implies that IP multicast packets are distributed using link-layer unicast transmissions, looking at the use of the word "copied" in this

section. Specifically in 6LoWPAN networks, this behavior conflicts with the requirement that IP multicast packets MUST be carried as link-layer 802.15.4 broadcast frames [RFC4944].

Assuming that link-layer unicast is indeed meant, this approach seems efficient only in a balanced, sparse tree network topology, or in situations where the fraction of nodes listening to a specific multicast IP address is low, or in duty cycled LLNs where link-layer broadcast is a very expensive operation.

4.3.1.3. RPL Routers with Non-RPL Hosts

Now we consider the case that hosts exist in a RPL network that are not RPL-aware themselves, but use link-local RPL routers for their IP connectivity. Note that the current RPL specification [I-D.ietf-roll-rpl] considers this case to be out of scope. However, it was suggested on the ROLL mailing list that RPL could potentially be run with non-RPL-aware hosts but that it is simply not specified yet. Such non-RPL hosts can't advertize their IP multicast groups of interest via RPL DAO messages as defined above. Therefore in that case MLD can be used for such advertizements (State Change Report messages), with all or a subset of RPL routers acting in the role of MLD Routers as defined in [RFC3810]. However, as the MLD protocol is not designed specifically for LLNs it may be a burden for the constrained RPL router nodes to run the full MLD protocol. Alternatives are therefore proposed in Section 4.5.1.

4.3.1.4. Trickle Multicast Forwarding

Trickle Multicast Forwarding [I-D.ietf-roll-trickle-mcast] is an IP multicast routing protocol suitable for LLNs, that uses the Trickle algorithm as a basis. It is a simple protocol in the sense that no topology maintenance is required. It can deal especially well with situations where the node density is a-priori unknown.

Nodes from anywhere in the LLN can be the multicast source, and nodes anywhere in the LLN can be multicast destinations.

Using Trickle Multicast Forwarding it is not required for IP multicast destinations (listeners) to announce their interest in a specific multicast IP address, e.g. by means of MLD. Instead, all multicast IP packets regardless of IP destination address are stored and forwarded by all routers. Because forwarding is always done by multicast, both hosts and routers will be able to receive all multicast IP packets. Routers that receive multicast packets they are not interested in, will only buffer these for a limited time until retransmission can be stopped as specified by the protocol. Hosts that receive multicast packets they are not interested in, will

discard multicast packets that are not of interest. Above properties seem to make Trickle especially efficient for cases where the multicast listener density is high and the number of distinct multicast groups relatively low.

4.3.1.5. Other Route-Over Methods

Other known IP multicast routing methods may be used, for example flooding or other to be defined methods suitable for LLNs. An important design consideration here is whether multicast listeners need to advertise their interest in specific multicast addresses, or not. If they do, MLD is a possible option but also protocol-specific means (as in RPL) is an option. See Section 4.5.1 for more efficient substitutes for MLD targeted towards a LLN context.

4.3.2. Single LLN with Backbone Topology

A LLN may be connected via a Border Router (e.g. 6LBR) to a backbone network, on which IP multicast listeners and/or sources may be present. This section analyzes cases in which IP multicast traffic needs to flow from/to the backbone, to/from the LLN.

4.3.2.1. Mesh-Under Multicast Routing

Because in a mesh routing network conceptually all nodes in the LLN are attached to a single link, a multicast IP packet originating in the LLN is typically delivered by the mesh routing algorithm to the 6LBR as well, although there is no guaranteed delivery. The 6LBR may be configured to accept all IP multicast traffic from the LLN and then may forward such packets onto its backbone link. Alternatively, the 6LBR may act in an MLD Router or MLD Snooper role on its backbone link and decide whether to forward a multicast packet or not based on information learnt from previous MLD Reports received on its backbone link.

Conversely, multicast packets originating on the backbone network will reach the 6LBR if either the backbone is a single link (LAN/VLAN) or IPv6 multicast routing is enabled on the backbone. Then, the 6LBR could simply forward all IP multicast traffic from the backbone onto the LLN. However, in practice this situation may lead to overload of the LLN caused by unnecessary multicast traffic. Therefore the 6LBR SHOULD only forward traffic that one or more nodes in the LLN have expressed interest in, effectively filtering inbound LLN multicast traffic.

To realize this "filter", nodes on the LLN may use MLD to announce their interest in specific multicast IP addresses to the 6LBR. One option is for the 6LBR to act in an MLD Router role on its LLN

interface. However, this may be too much of a "burden" for constrained nodes. Light-weight alternatives for MLD are discussed in Section 4.5.1.

4.3.2.2. RPL Multicast Routing

For RPL routing within the 6LoWPAN, we first consider the case of an IP multicast source on the backbone network with one or more IP multicast listeners on the RPL LLN. Typically, the 6LBR would be the root of a DODAG so that the 6LBR can easily forward the IP multicast packet received on its backbone interface to the right RPL nodes in the LLN down along this DODAG (based on previously DAO-advertized destinations).

Second, a multicast source may be in the RPL LLN and listeners may be both on the LLN and on the backbone. For this case RPL defines that the multicast packet will propagate both up and down the DODAG, eventually reaching the DODAG root (typically a 6LBR) from which the packet can be routed onto the backbone in a manner specified in the previous section.

4.3.2.3. RPL Routers with Non-RPL Hosts

For the case that a RPL LLN contains non-RPL hosts, the solutions from the previous section can be used if in addition RPL routers implement MLD or "MLD like" functionality similar to as described in Section 4.3.1.3.

4.3.2.4. Trickle Multicast Forwarding

First, we consider the case of an IP multicast source node on the LLN (where all 6LRs support Trickle Multicast Forwarding) and IP multicast listeners that may be on the LLN and on the backbone. As Trickle will eventually deliver multicast packets also to a 6LBR, which acts as a Trickle Multicast router as well, the 6LBR can then forward onto the backbone in the ways described earlier in Section 4.3.2.1.

Second, for the case of an IP multicast source on the backbone and multicast listeners on both backbone and/or LLN, the 6LBR needs to forward multicast traffic from the backbone onto the LLN. Here, the aforementioned problem (Section 4.3.2.1) of potentially overloading the LLN with unwanted backbone IP multicast traffic appears again.

A possible solution to this is (again) to let multicast listeners advertize their interest using MLD as described in Section 4.3.2.1 or to use an MLD alternative suitable for LLNs as described in Section 4.5.1. However, following this approach requires possibly an

extension to Trickle Multicast Forwarding: the protocol should ensure that MLD-advertized information is somehow communicated to the 6LBR, possibly over multiple hops. MLD itself supports link-local communication only.

4.3.2.5. Other Route-Over Methods

For other multicast routing methods used on the LLN, there are similar considerations to the ones in sections above: the strong need to filter IP multicast traffic coming into the LLN, the need for reporting multicast listener interest (e.g. with MLD or a to-be-defined MLD alternative) by constrained (6LoWPAN) nodes, and the need for LLN-internal routing as identified in the previous section such that the MLD communicated information can reach the 6LBR to be used there in multicast traffic filtering decisions.

4.3.3. Multiple LLNs with Backbone Topology

Now the case of a single backbone network with two or more LLNs attached to it via 6LBRs is considered. For this case all the considerations and solutions of the previous section can be applied.

For the specific case that a source on a backbone network has to send to a very large number of destination located on many LLNs, the use of IGMP/MLD Proxying [RFC4605] with a leaf IGMP/MLD Proxy located in each 6LBR may be useful. This method only is defined for a tree topology backbone network with the IP multicast source at the root of the tree.

4.3.4. LLN(s) with Multiple 6LBRs

[TBD: an LLN with multiple 6LBRs may require some additional consideration. Any need to synchronize mutually on multicast listener information?]

4.3.5. Conclusions

For all network topologies that were evaluated, CoAP group communication can be in principle supported with IP Multicast, making use of existing protocols. For the case of Trickle Multicast Forwarding, it appears that an addition to the protocol is required such that information about multicast listeners can be distributed towards the 6LBR. Opportunities were identified for an "MLD-like" or "MLD-lightweight" protocol specifically suitable for LLNs, which should interwork with regular MLD on the backbone network. Such MLD variants are further analyzed in Section 4.5.1.

4.4. HTTP/CoAP Interworking Aspects

The topic of HTTP unicast to CoAP multicast request proxying is treated in [I-D.castellani-core-http-mapping]. [TBD: only if needed more information will be added here in the future.]

4.5. Implementation Considerations

In this section various implementation aspects are considered such as required protocol implementations, additional functionality of the 6LBR and backbone network equipment.

4.5.1. MLD Implementation on LLNs

In previous sections, it was mentioned that the MLDv2 protocol [RFC3810] may be too costly for use in a LLN. MLD relies on periodic link-local multicast operations to maintain state. Also it is optimized to fairly dynamic situations where multicast listeners may come and go over time. Such dynamic situations are less frequently found in typical LLN use cases such as building control, where multicast group membership can remain constant over longer periods of time (e.g. months) after commissioning.

Hence, a viable strategy is to implement a subset of MLD functionality in 6LoWPAN nodes which is just enough for the required functionality. A first option is that 6LoWPAN Routers, like MLD Snooters, passively listen to MLD State Change Report messages and handle the learnt ("snooped") IP multicast destinations in the way defined by the multicast routing protocol they are running (e.g. for RPL, Routers advertize these destinations using DAO messages).

A second option is to use MLD as-is but adapt the recommended parameter values such that operation on a LLN becomes more efficient.

A third option is to standardize a new protocol, taking a subset of MLD functionality into a "MLD for 6LoWPAN" protocol to support constrained nodes optimally.

A fourth option is now presented, which seems attractive in that it minimizes standardization, implementation and network communication overhead all at the same time. This option is to specify a new Multicast Listener Option (MLO) as an addition to the 6LoWPAN-ND [I-D.ietf-6lowpan-nd] protocol communication that is anyway ongoing between a 6LoWPAN host and router(s). This MLO is preferably designed to be maximally similar to the Address Registration Option (ARO), which minimizes the need for additional program code on constrained nodes. With an MLO, instead of registering a unicast IP address, a host "registers" its interest in a multicast IP address.

Unlike ARO, multiple MLO can be used in the same ND packet. A registration period is also defined just like in the ARO. MLO allows a host to persistently register as a listener to IP multicast traffic and to avoid the overhead of periodic multicast communication which is required for full MLD.

[TBD: consider what aspects are needed/not needed for CoAP/LLN applications. Will MLDv1 suffice? What to do with options like 'source specific' and include/exclude. Source-specific can also be dealt with at the destination host by filtering? Do we need limits on number of records per packet? Do we need a higher MLD reliability setting - see the parameters in the MLD RFC]

4.5.2. 6LBR Implementation

To support mixed backbone/LLN scenarios in CoAP group communication, it is RECOMMENDED that a 6LowPAN Border Router (6LBR) will act in an MLD Router role on the backbone link. If this is not possible then the 6LBR SHOULD be configured to act as an MLD Multicast Address Listener and/or MLD Snooper on the backbone link.

4.5.3. Backbone IP Multicast Infrastructure

For corporate/professional applications, most routing and switching equipment that is currently on the market is IPv6 capable. For that reason backbone infrastructure operating IPv4 only is considered out of scope in this document, at least for the backbone network segment(s) where IP multicast destinations are present. What is still in scope is for example an IPv4-only HTTP client that wants to send a group communication message via a HTTP-CoAP proxy as considered in [I-D.castellani-core-http-mapping].

The availability of, and requirements for, IP multicast support may depend on the specific installation use case. For example, the following cases may be relevant for new IP based building control installations:

1. System deployed on existing IP (Ethernet/WiFi/...) infrastructure, shared with existing IP devices (PCs)
2. Newly designed & deployed IP (Ethernet/WiFi/...) infrastructure, to be shared with other IP devices (PCs)
3. Newly designed & deployed IP (Ethernet/WiFi/...) infrastructure, exclusively used for building control.

Besides physical separation the building control backbone can be separated from regular (PC) infrastructure by using a different VLAN. A typical corporate installation will have many LAN switches and/or routing switches, which pass through IP multicast traffic but on the other hand do not support acting in the Router role of MLD/IGMP. Perhaps for case 2) and 3) above it is acceptable to add a MLD/IGMP

capable router somewhere in the network, while for case 1) this may not be the case.

[TBD: consider the influence of WiFi based backbone networks. What if 6LBRs are at the same time also WiFi routers? What if 6LBRs have an Ethernet connection to legacy WiFi routers? Check if equivalent with Ethernet backbone.]

5. Security Considerations

Security for group communications at the IP level has been studied extensively in the IETF MSEC (Multicast Security) WG, and to a lesser extent in the IRTF SAMRG (Scalable Adaptive Multicast Research Group). In particular, [RFC3740], [RFC5374] and [RFC4046] are very instructive. A set of requirements for securing group communications in CoAP were derived from a study of these previous investigations as well as understanding of CoAP specific needs. These are listed below.

Note that some of the requirements are marked optional. This means that, depending on the use case, these may be required or not. For this purpose each use case can be associated to a security profile as specified in [I-D.garcia-core-security]. The security profile prescribes what requirements should be taken into account for this profile. A mapping of these requirements to these profiles has not yet been done.

REQ1- Group communications data encryption: Important CoAP group communications shall be encrypted (using a group key) to preserve confidentiality. It shall also be possible to send CoAP group communications in the clear (i.e. unencrypted) for low value data.

REQ2- Group communications source data authentication: Important CoAP group communications shall be authenticated by verifying the source of the data (i.e. that it was generated by a given and trusted group member). It shall also be possible to send unauthenticated CoAP group communications for low value data.

REQ3- Group communications limited data authentication: Less important CoAP group communications shall be authenticated by simply verifying that it originated from one of the group members (i.e. without explicitly identifying the source node). This is a weaker requirement (but simpler to implement) than REQ2. It shall also be possible to send unauthenticated CoAP group communications for low value data.

REQ4- Group key management: There shall be a secure mechanism to

manage the cryptographic keys (e.g. generation and distribution) belonging to the group; the state (e.g. current membership) associated with the keys; and other security parameters.

REQ5- Use of Multicast IPsec: The CoAP protocol [I-D.ietf-core-coap] allows IPsec to be used as one option to secure CoAP. If IPsec is used as a way to security CoAP communications, then multicast IPsec [RFC5374] should be used for securing CoAP group communications.

REQ6- Independence from underlying routing security: CoAP group communication security shall not be tied to the security of underlying routing and distribution protocols such as PIM [RFC4601] and RPL [I-D.ietf-roll-rpl]. Insecure or inappropriate routing (including IP multicast routing) may cause loss of data to CoAP but will not affect the authenticity or secrecy of CoAP group communications.

REQ7- Interaction with HTTPS: The security scheme for CoAP group communications shall account for the fact that it may need to interact with HTTPS (Hypertext Transfer Protocol Secure) when a transaction involves a node in the general Internet (non-constrained network) communicating via a HTTP-CoAP proxy.

6. IANA Considerations

This document makes no request of IANA.

7. Conclusions

Three solutions for enabling CoAP group communications have been discussed.

Unreliable IP multicast as outlined in Section 3.3 is recommended to be adopted as the base solution for CoAP Group Communication on LLNs. This approach requires no standards changes to the IP multicast suite of protocols and it provides interoperability with IP multicast group communication on unconstrained backbone networks.

The proposals for group communication described in this draft should be considered for incorporation into the overall CoAP protocol specification.

8. Acknowledgements

Thanks to Peter Bigot, Carsten Bormann, Anders Brandt, Angelo

Castellani, Guang Lu, Salvatore Loreto, Kerry Lynn, Dale Seed, Zach Shelby, Peter van der Stok, and Juan Carlos Zuniga for their helpful comments and discussions that have helped shape this document.

9. References

9.1. Normative References

- [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.
- [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.
- [RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [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.
- [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.
- [RFC4286] Haberman, B. and J. Martin, "Multicast Router Discovery", RFC 4286, December 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM):

Protocol Specification (Revised)", RFC 4601, August 2006.

- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, November 2008.
- [RFC5771] Cotton, M., Vegoda, L., and D. Meyer, "IANA Guidelines for IPv4 Multicast Address Assignments", BCP 51, RFC 5771, March 2010.

9.2. 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.eggert-core-congestion-control]
Eggert, L., "Congestion Control for the Constrained Application Protocol (CoAP)", draft-eggert-core-congestion-control-01 (work in progress), January 2011.
- [I-D.ietf-6lowpan-hc]
Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams in Low Power and Lossy Networks (6LoWPAN)", draft-ietf-6lowpan-hc-15 (work in progress), February 2011.
- [I-D.ietf-6lowpan-nd]
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low Power and Lossy Networks (6LoWPAN)", draft-ietf-6lowpan-nd-17 (work in progress), June 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.ietf-core-observe]
Hartke, K. and Z. Shelby, "Observing Resources in CoAP",
draft-ietf-core-observe-02 (work in progress), March 2011.
- [I-D.shelby-core-coap-req]
Shelby, Z., Stuber, M., Sturek, D., Frank, B., and R.
Kelsey, "CoAP Requirements and Features",
draft-shelby-core-coap-req-02 (work in progress),
October 2010.
- [I-D.vanderstok-core-bc]
Stok, P. and K. Lynn, "CoAP Utilization for Building
Control", draft-vanderstok-core-bc-04 (work in progress),
July 2011.
- [I-D.castellani-core-http-mapping]
Castellani, A., Loreto, S., Rahman, A., Fossati, T., and
E. Dijk, "Best practices for HTTP-CoAP mapping
implementation", draft-castellani-core-http-mapping-01
(work in progress), July 2011.
- [I-D.garcia-core-security]
Garcia-Morchon, O., Keoh, S., Kumar, S., Hummen, R., and
R. Struik, "Security Considerations in the IP-based
Internet of Things", draft-garcia-core-security-02 (work
in progress), July 2011.
- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., Brandt, A., Clausen, T., Hui, J.,
Kelsey, R., Levis, P., Pister, K., Struik, R., and J.
Vasseur, "RPL: IPv6 Routing Protocol for Low power and
Lossy Networks", draft-ietf-roll-rpl-19 (work in
progress), March 2011.
- [I-D.ietf-roll-trickle-mcast]
Hui, J. and R. Kelsey, "Multicast Forwarding Using
Trickle", draft-ietf-roll-trickle-mcast-00 (work in
progress), April 2011.
- [ID.goland-http-udp]
Goland, Y., "Multicast and Unicast UDP HTTP Messages",
1999,
<<http://tools.ietf.org/html/draft-goland-http-udp-01>>.

- [STUDY1] Lao, L., Cui, J., Gerla, M., and D. Maggiorini, "A Comparative Study of Multicast Protocols: Top, Bottom, or In the Middle?", 2005, <http://www.cs.ucla.edu/NRL/hpi/AggMC/papers/comparison_gi_2005.pdf>.
- [STUDY2] Banerjee, B. and B. Bhattacharjee, "A Comparative Study of Application Layer Multicast Protocols", 2001, <<http://wmedia.grnet.gr/P2PBackground/a-comparative-study-ofALM.pdf>>.

Authors' Addresses

Akbar Rahman (editor)
InterDigital Communications, LLC
Email: Akbar.Rahman@InterDigital.com

Esko Dijk (editor)
Philips Research
Email: esko.dijk@philips.com

CoRE
Internet-Draft
Intended status: Informational
Expires: November 3, 2011

Z. Shelby
Sensinode
M. Garrison Stuber
Itron
D. Sturek
Pacific Gas & Electric
B. Frank
Tridium, Inc
R. Kelsey
Ember
May 2, 2011

CoAP Requirements and Features
draft-shelby-core-coap-req-04

Abstract

This document considers the requirements for the design of the Constrained Application Protocol (CoAP). The goal of the document is to provide a basis for protocol design and related discussion.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on November 3, 2011.

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 BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. CoAP Requirements	3
3. Applicability	5
3.1. Energy Applications	5
3.2. Building Automation	6
3.3. General M2M Applications	7
4. Conclusions	7
5. Security Considerations	7
6. IANA Considerations	7
7. Acknowledgments	7
8. References	8
8.1. Normative References	8
8.2. Informative References	8
Authors' Addresses	8

1. Introduction

The use of web services on the Internet has become ubiquitous in most applications, and depends on the fundamental Representational State Transfer (REST) architecture of the web. The proposed Constrained RESTful Environments (CoRE) working group 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). One of the main goals of CoRE is to design a generic RESTful protocol for the special requirements of this constrained environment, especially considering energy and building automation applications. The result of this work should be a Constrained Application Protocol (CoAP) which easily translates to HTTP for integration with the web while meeting specialized requirements such as multicast support, very low overhead and simplicity.

This document first analyzes the requirements for CoAP from the proposed charter and related application requirement drafts in Section 2. The applicability of these requirements to energy, building automation and general M2M applications is considered in Section 3.

2. CoAP Requirements

The following requirements for CoAP have been identified in the proposed charter of the working group (Feb 13, 2010 version), in the 6lowapp problem statement [I-D.bormann-6lowpan-6lowapp-problem], or in the application specific requirement documents. This section is not meant to introduce new requirements, only to summarize the requirements from other sources. The requirements relevant to CoAP can be summarized as follows:

- REQ1: CoRE solutions must be of appropriate complexity for use by nodes have limited code size and limited RAM (e.g. microcontrollers used in low-cost wireless devices typically have on the order of 64-256K of flash and 4-12K of RAM).
[charter], [I-D.sturek-6lowapp-smartenergy]
- REQ2: Protocol overhead and performance must be optimized for constrained networks, which may exhibit extremely limited throughput and a high degree of packet loss. For example, multihop 6LoWPAN networks often exhibit application throughput on the order of tens of kbits/s with a typical payload size of 70-90 octets after transport layer headers.
[charter]

- REQ3: The ability to deal with sleeping nodes. Devices may be powered off at any point in time but periodically "wake up" for brief periods of time. [charter], [I-D.sturek-6lowapp-smartenergy], [I-D.gold-6lowapp-sensei]
- REQ4: Protocol must support the caching of recent resource requests, along with caching subscriptions to sleeping nodes. [charter]
- REQ5: Must support the manipulation of simple resources on constrained nodes and networks. The architecture requires push, pull and a notify approach to manipulating resources. CoAP will be able to create, read, update and delete a Resource on a Device. [charter], [I-D.sturek-6lowapp-smartenergy], [I-D.martocci-6lowapp-building-applications], [I-D.gold-6lowapp-sensei]
- REQ6: The ability to allow a Device to publish a value or event to another Device that has subscribed to be notified of changes, as well as the way for a Device to subscribe to receive publishes from another Device. [charter]
- REQ7: Must define a mapping from CoAP to a HTTP REST API; this mapping will not depend on a specific application and must be as transparent as possible using standard protocol response and error codes where possible. [charter], [I-D.sturek-6lowapp-smartenergy], [I-D.gold-6lowapp-sensei]
- REQ8: A definition of how to use CoAP to advertise about or query for a Device's description. This description may include the device name and a list of its Resources, each with a URL, an interface description URI (pointing e.g. to a Web Application Description Language (WADL) document) and an optional name or identifier. The name taxonomy used for this description will be consistent with other IETF work, e.g. draft-cheshire-dnsext-dns-sd. [charter]
- REQ9: CoAP will support a non-reliable IP multicast message to be sent to a group of Devices to manipulate a resource on all the Devices simultaneously [charter]. The use of multicast to query and advertise descriptions must be supported, along with the support of unicast responses [I-D.sturek-6lowapp-smartenergy].

- REQ10: The core CoAP functionality must operate well over UDP and UDP must be implemented on CoAP Devices. There may be optional functions in CoAP (e.g. delivery of larger chunks of data) which if implemented are implemented over TCP. [charter], [I-D.sturek-6lowapp-smartenergy], [I-D.martocci-6lowapp-building-applications]
- REQ11: Reliability must be possible for unicast application layer messages over UDP [I-D.sturek-6lowapp-smartenergy].
- REQ12: Latency times should be minimized of the Home Area Network (HAN), and ideally a typical exchange should consist of just a single request/response exchange. [I-D.sturek-6lowapp-smartenergy]
- REQ13: A subset of Internet media types must be supported. [I-D.sturek-6lowapp-smartenergy], [I-D.gold-6lowapp-sensei]
- REQ14: Consider operational and manageability aspects of the protocol and at a minimum provide a way to tell if a Device is powered on or not. [charter]

3. Applicability

This sections looks at the applicability of the CoAP features for energy, building automation and other machine-to-machine (M2M) applications.

3.1. Energy Applications

Rising energy prices, concerns about global warming and energy resource depletion, and societal interest in more ecologically friendly living have resulted in government mandates for Smart Energy solutions. In a Smart Energy environment consumers of energy have direct, immediate access to information about their consumption, and are able to take action based on that information. Smart Energy systems also allow device to device communication to optimize the transport, reliability, and safety of energy delivery systems. While often Smart Energy solutions are electricity-centric, i.e. Smart Grid, gas and water are also subject to the same pressures, and can benefit from the same technology.

Smart Energy Transactions typically include the exchange of current consumption information, text messages from providers to consumers, and control signals requesting a reduction in consumption. Advanced features such as billing information, energy prepayment transactions, management of distributed energy resources (e.g. generators and

photo-voltaics), and management of electric vehicles are also being developed.

Smart Energy benefits from Metcalfe's Law. The more devices that are part of a smart energy network within the home or on the grid, the more valuable it becomes. Showing a consumer how much energy they are using is useful. Combining that with specific information about their major appliances, and enabling them to adjust their consumption based on current pricing and system demand is much much more powerful. To do this however requires a system that is resilient, low cost, and easy to install. In many areas this is being done with systems built around IEEE 802.15.4 radios. In the United States, there are over 30 million electric meters that will be deployed with these radios. These radios will be combined to form a mesh network, enabling Smart Energy communication within the home. The maximum packet size for IEEE 802.15.4 is only 127 bytes. Additionally, there is the well known issue of how TCP manages congestion working sub-optimally over wireless networks. IEEE 802.15.4 is ideal for these applications because of its low cost and its support for battery powered devices; however, it is not as well suited for heavier protocols like HTTP. These technical issues with IEEE 802.15.4 networks combined with a desire to facilitate broader compatibility, makes a protocol like CoAP desirable. Its REST architecture will allow seamless compatibility with the rest of the Internet, allowing it to be easily integrated with web browsers and web-based service providers, while at the same time being appropriately sized for the low-cost networks necessary for its success.

3.2. Building Automation

Building automation applications were analyzed in detail including use cases in [I-D.martocci-6lowapp-building-applications]. Although many of the embedded control solutions for building automation make use of industry-specific application protocols like BACnet over IP, there is a growing use of web services in building monitoring, remote control and IT integration. The OASIS oBIX standard [ref] is one example of the use of web services for the monitoring and interconnection of heterogeneous building systems. Several of the CoAP requirements have been taken from [I-D.martocci-6lowapp-building-applications]. The resulting features should allow for peer-to-peer interactions as well as node-server request/response and push interactions for monitoring and some control purposes. For building automation control with very strict timing requirements using e.g. multicast, further features may be required on top of CoAP.

3.3. General M2M Applications

CoAP provides a natural extension of the REST architecture into the domain of constrained nodes and networks, aiming at requirements from automation applications in energy and building automation. A very wide range of machine-to-machine (M2M) applications have similar requirements to those considered in this document, and thus it is foreseen that CoAP may be widely applied in the industry. One standardization group considering a general M2M architecture and API is the ETSI M2M TC, which considers a wide range of applications including energy. Another group developing solutions for general embedded device control is the OASIS Device Profile Web Services (DPWS) group. The consideration of DPWS over 6LoWPAN is available in [I-D.moritz-6lowapp-dpws-enhancements].

4. Conclusions

This document analyzed the requirements associated with the design of the foreseen Constrained Application Protocol (CoAP). The identified requirements of CoAP are considered for energy, building automation and M2M applications. This document is meant to serve as a basis for the design of the CoAP protocol and relevant discussion.

5. Security Considerations

The CoAP protocol will be designed for use with e.g. (D)TLS or object security. A protocol design should consider how integration with these security methods will be done, how to secure the CoAP header and other implications.

6. IANA Considerations

This draft requires no IANA consideration.

7. Acknowledgments

Thanks to Cullen Jennings, Guido Moritz, Peter Van Der Stok, Adriano Pezzuto, Lisa Dussealt, Gilbert Clark, Salvatore Loreto, Alexey Melnikov and Bob Dolin for helpful comments and discussions.

8. References

8.1. Normative References

- [I-D.gold-6lowapp-sensei]
Gold, R., Krco, S., Gluhak, A., and Z. Shelby, "SENSEI 6lowapp Requirements", draft-gold-6lowapp-sensei-00 (work in progress), October 2009.
- [I-D.martocci-6lowapp-building-applications]
Martocci, J. and A. Schoofs, "Commercial Building Applications Requirements", draft-martocci-6lowapp-building-applications-00 (work in progress), October 2009.
- [I-D.sturek-6lowapp-smartenergy]
Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requirements for 6LowApp", draft-sturek-6lowapp-smartenergy-00 (work in progress), October 2009.

8.2. Informative References

- [I-D.bormann-6lowpan-6lowapp-problem]
Bormann, C., Sturek, D., and Z. Shelby, "6LowApp: Problem Statement for 6LoWPAN and LLN Application Protocols", draft-bormann-6lowpan-6lowapp-problem-01 (work in progress), July 2009.
- [I-D.moritz-6lowapp-dpws-enhancements]
Moritz, G., "DPWS for 6LoWPAN", draft-moritz-6lowapp-dpws-enhancements-00 (work in progress), December 2009.

Authors' Addresses

Zach Shelby
Sensinode
Kidekuja 2
Vuokatti 88600
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

Michael Garrison Stuber
Ittron
2111 N. Molter Road
Liberty Lake, WA 99025
U.S.A.

Phone: +1.509.891.3441
Email: Michael.Stuber@itron.com

Don Sturek
Pacific Gas & Electric
77 Beale Street
San Francisco, CA
USA

Phone: +1-619-504-3615
Email: d.sturek@att.net

Brian Frank
Tridium, Inc
Richmond, VA
USA

Phone:
Email: brian.tridium@gmail.com

Richard Kelsey
Ember
47 Farnsworth Street
Boston, MA 02210
U.S.A.

Phone: +1.617.951.1201
Email: richard.kelsey@ember.com

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 preferable 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 its 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 leaf 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 responsible 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 specify 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 enumerate 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: ffile::148. The accompanying SRV, AAAA, and TXT RR become:

```
_all_light.04.b8 SRV  all_light.o4.b8.example.com Port-z
                    AAAA ffile::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 exists in the case of multi-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 lml, 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 fffe::148
TXT if=ZigBee path=/light
_light_1.04.b8 SRV lml.o4.b8.example.com Port-z
AAAA fdfe::1234
TXT if=ZigBee path=/light/1
_light_2.04.b8 SRV lml.o4.b8.example.com Port-z
AAAA fdfe::1234
TXT if=ZigBee path=/light/2
_light_3.04.b8 SRV lml.o4.b8.example.com Port-z
AAAA fdfe::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-dnsexst-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  fdfe:: 1241
                        TXT  path=/os if=ZigBee
id-1._temp_sensor       SRV  proxy.o4.b8.example.com Port-x
                        AAAA  fdfe:: 1241
                        TXT  path=/ts if=ZigBee
sl-ts.o4.b8.example.com AAAA  fdfe::1242
sl-os.o4.b8.example.com AAAA  fdfe::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

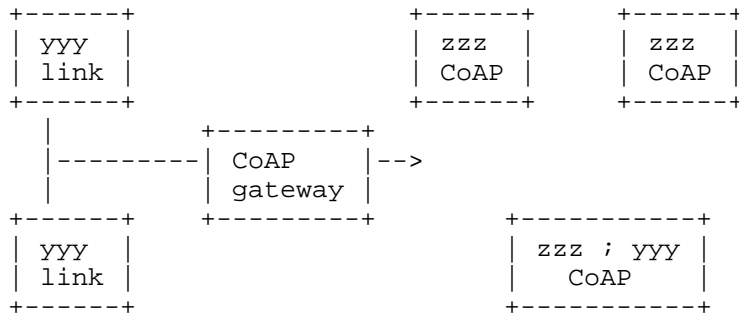


Figure 1: network with multiple representation standards

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).

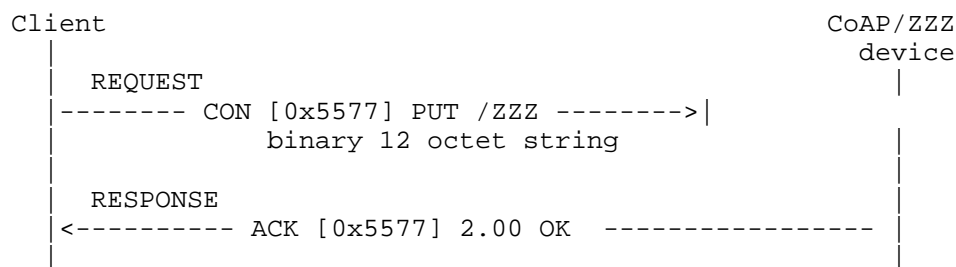


Figure 2: Sending a ZZZ command with CoAP to CoAP/ZZZ device

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.

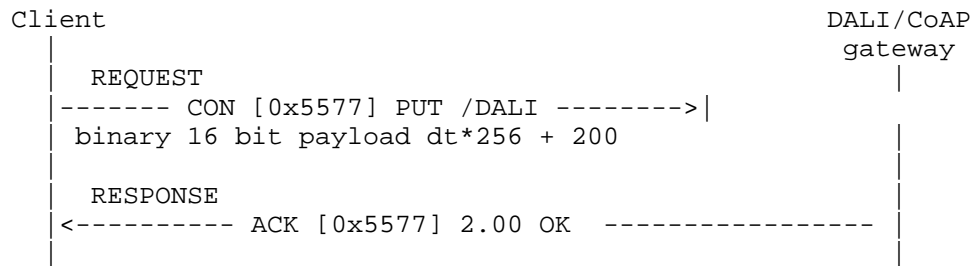


Figure 3: Sending a DALI setting with CoAP to CoAP/DALI gateway

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),

resource, and attribute.

From bc-04 to bc-05

- extended and corrected examples for multi-function devices
- 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-dnsexst-site-mdns]

Lynn, K. and D. Sturek, "Extended Multicast DNS",
draft-lynn-dnsexst-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](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

