Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2012

T. Fossati
KoanLogic
P. Giacomin
Freelance
S. Loreto
Ericsson

March 10, 2012

Publish and Monitor Options for CoAP
draft-fossati-core-publish-monitor-options-01

Abstract

This memo defines two additional Options for the Constrained Application Protocol (CoAP) especially targeted at sleepy sensors: Publish and Monitor.

The Publish Option enables opportunistic updates of a given resource state, by temporarily delegating the authority of the Publish'ed resource to a Proxy node. The whole process is driven by the (sleepy) origin -- which may actually never need to listen.

The Monitor Option complements the typical Observe pattern, enabling the tracking of a resource hosted by a node sleeping most of the time, by taking care of establishing and maintaining an Observe relationship with the (sleepy) origin on behalf of the (sleepy) client.

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 September 11, 2012.

Copyright Notice

Table of Contents

1. Introduction

The proposal described in this memo covers the following use case:

> a node N, which is sleeping most of the time, depends one or more resources hosted at another sleepy node M. In cases as such, the probability of an empty intersection between their respective wake periods is very high, making it hard for the two to synchronize.

In this scenario, using the basic observe [I-D.ietf-core-observe] functionality is not enough, as it could lead to lost state updates in case N is offline while M pushes its notifications; further, the observation may never bootstrap since its initialization needs both client and origin awake at the same time.

This memo introduces two extensions to the Proxy caching functionality that give the Proxy an explicit mediation role in the sleepy-to-sleepy CoAP [I-D.ietf-core-coap] communication.

1.1. Requirements Language and Motivation

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].

This specification makes use of the following terminology:

Sleepy Device: a sensor/actuator (usually battery operated) that powers down its radio beyond the normal radio duty cycle in order to save energy.

and tries to provide an in-protocol solution for the requirement REQ3 stated in [I-D.shelby-core-coap-req]:

> The ability to deal with sleeping nodes. Devices may be powered down at any point in time but periodically "wake up"

for brief periods of time.

2. Options

No.	C/E	Name	Format	Length	Default
YY	Critical	Publish	uint	1 B	0x2
XX	Critical	Monitor	(none)	0 B	(none)

2.1. Publish Option

The Publish Option enables the sleepy origin to temporarily (i.e. for a specified "lease" time) delegate the authority of one of its hosted resources to a Proxy node that will start to behave as the origin for the Publish'ed resource. This allows a sleepy sensor to use the Proxy as the rendezvous point for one-way sleepy to sleepy signaling.

2.1.1. Publishing a Resource

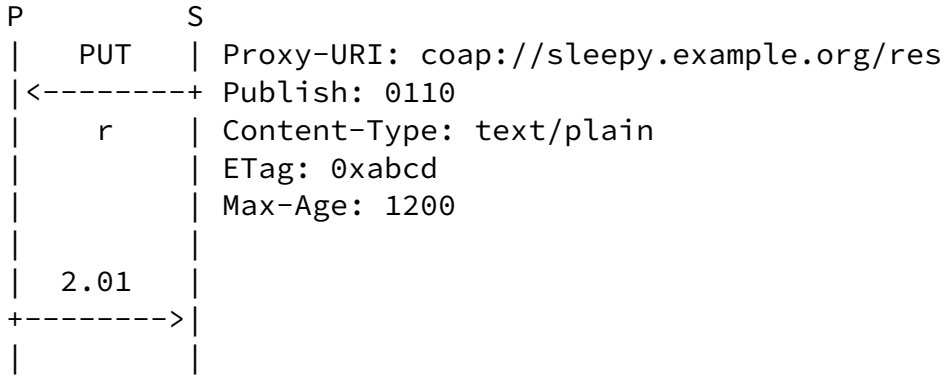


Figure 1

The origin server publishes one of its hosted resources, specified by the enclosed Proxy-URI, by PUT'ing it to the Proxy with a Publish Option attached. The Publish Option value specifies the CoAP methods that clients are allowed to use on the resource (see Section 2.1.4).

The example in Figure 1 shows a delegation where the GET and PUT

methods are allowed while POST and DELETE are explicitly prohibited, meaning that the resource can be read and updated by clients, but it can't be deleted nor created.

The Proxy, which is voluntarily charged by the resource owner to act as the delegated origin for the "lease" time specified by Max-Age, replies with a 2.01 if the authority transfer has succeeded. An exact duplicate of the submitted representation is created, and from now on it can be accessed using the original URI provided that clients go through the delegated Proxy. If the Publish operation does not succeed, the origin transfer fails, and an appropriate response code is returned.

An ETag MAY be supplied as a metadata to be included in responses involving the Publish'ed representation. If no Max-Age is given, a default of 3600 seconds MUST be assumed. The Max-Age value, either implicit or explicit, determines the lifetime of the origin delegation. When the Max-Age value is elapsed, the Proxy MUST delete the Publish'ed resource value and fall back to its usual proxying

function.

The Publish Option is critical and MUST be present in the request only. If the Proxy does not recognize it, a 4.02 (Bad Option) MUST be returned to the client. If the option value is not correctly formatted (see Section 2.1.4), a 4.00 (Bad Request) MUST be returned to the client.

It is sufficient for any client wishing to access the resource to do so using the Proxy node that, following the Publish operation, will start behaving a the origin, satisfying requests on behalf of the sleeping node.

The Proxy MUST save the identity of the resource Publish'er in order to distinguish "maintainance" operations such as update and explicit deletion, from "regular" access to the published resource by clients.

An interesting outcome of this communication strategy is that the sleepy origin may really never need to listen on its radio interface.

2.1.2. Updating a Resource

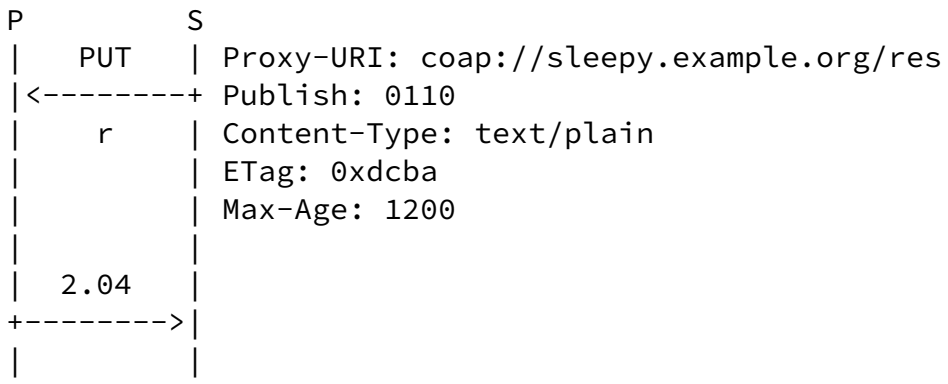
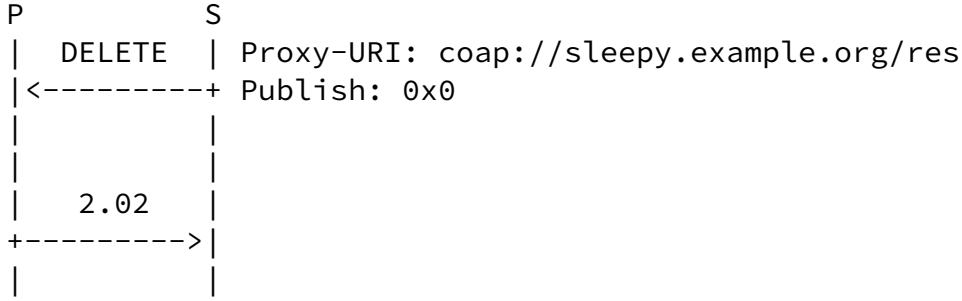


Figure 2

In order to update the delegated resource state, the sleepy node shall send the very same request to the Proxy, which in turn replies with a 2.04 (Changed) status code in case the update operation has succeeded, or an appropriate error code in case it fails.

2.1.3. Unpublishing a Resource



The delegation of a given resource can be explicitly revoked by the real origin at any time before the lease time expires, by issuing a DELETE request to the Proxy hosting the resource duplicate with a Publish Option with value 0x0.

On successful deletion of the delegation a 2.02 (Deleted) response code is returned by the Proxy.

2.1.4. Value Format

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|C R U D 0 0 0 0|
+-+-+-+-+-+-+-+-+
```

Each of the first 4 bits is a flag field indicating whether the associated CoAP method (respectively: POST, GET, PUT and DELETE) is allowed on the Publish'ed resource. The remaining 4 bits are unused and MUST be set to 0.

In case the value is missing, the default is assumed to be 0x2, i.e. the resource is read-only.

An all-0 value is used to explicitly revoke the delegation (see Section 2.1.3.)

If the delegated Proxy receives a request with a method that is not compatible with the supplied mask, it MUST respond with a 4.05 (Method Not Allowed) response code.

2.1.5. Discovery

2.1.5.1. Publishing the /.well-known/core Resource

The [I-D.ietf-core-link-format] has no explicit text about "well-known" discovery of devices through a Proxy, nor about the cacheability rules for such resource. Even if it seems reasonable to

assume that the /.well-known/core URI is both query-able and cacheable through a Proxy, on the contrary the situation is not very much so.

In fact, since the "well-known" interface relies on the resource origin being implicitly defined by the source address of the UDP packet carrying the response, quering the "well-known" interface (either unicast or multicast) through a Proxy-URI has little hope to

be fully functional. The (ab)use of a an implicit L3 locator as the identifier of the resource authority makes "well-known" discovery generally incompatible with Proxy mediated communication, unless each target URI in a link is given as a URI and not as a relative-ref (section 4.1 of [RFC3986]).

Consequently, in this proposal we assume that the /.well-known/core of a sleepy node can be Publish'ed if and only if the target URI in the each link is not a relative-ref.

Its registration is the same as in Figure 1, but the Proxy MAY need to treat it in a way that is slightly different from other "normal" delegated resources. In fact, while delegation is in place (i.e. the lease period is not elapsed, and neither explicit revocation has happened) the Proxy MAY be able to respond to filtered queries (section 4.1 of [I-D.ietf-core-link-format]) regarding the Publish'ed /.well-known/core.

2.1.5.2. Resource Directory

Given the strong requirement on the link formatting given in Section 2.1.5.1, it could be preferable (or even necessary) to use the Resource Directory [I-D.shelby-core-resource-directory] as a means of delegating the discovery of the resources hosted at a sleepy node.

This can be done either by the sleepy node, or automatically by the delegated Proxy when a Publish request is received.

[[Automatic push to RD: check it out]]

2.2. Monitor Option

The Monitor Option is a variant of the Observe Option that is aimed at solving some issues that may occur when sleepy sensors are involved.

Suppose that the resource of interest is not in cache, and a sleepy endpoint wants to Observe it through the Proxy. If the origin of the requested resource is sleeping at the time the observation is

requested, the requesting node gets an error, and may need to stay

awake and retry until the target node gets ready -- which is clearly not an option in case the sensor has a very small duty cycle.

The Monitor Option is used to ask a Proxy to keep a given resource fresh by observing it, while the requesting node is sleeping. Thus the sleepy sensor can possibly get the latest representation published by the monitored resource when it wakes up, even if the origin is sleeping -- and was sleeping at the time the Monitor has been requested.

The Monitor Option is critical and MUST be present in the request only. If the Proxy does not recognize it, a 4.02 (Bad Option) MUST be returned to the client.

2.2.1. Public Monitor Registration

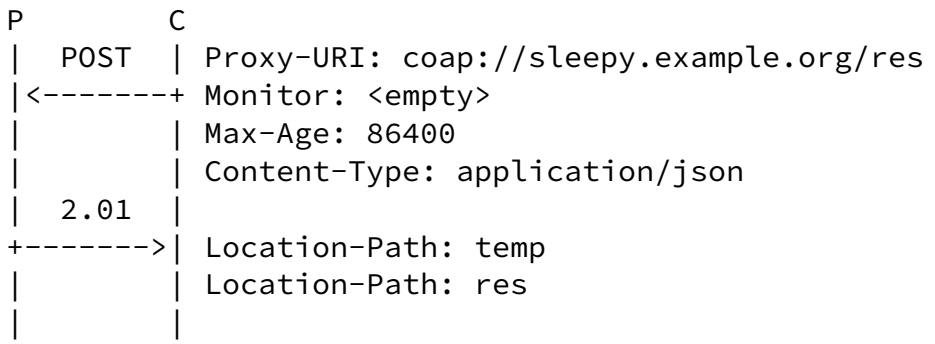


Figure 3

The client POST's the resource to be monitored, identified by the Proxy-URI. The request message contains an empty Monitor Option, and possibly specifies a TTL (i.e. an implicit de-registration indication) for the monitor through Max-Age. One or more content types for the acceptable representations of the resource are optionally specified via the Accept option. In case no TTL is supplied, a default value of 3600 seconds is assumed.

The operation creates a "monitor" resource at the Proxy, that MUST maintain a fresh carbon copy of one or more representations of the requested resource depending on the supplied Content-Type's. For convenience, multiple "monitor" resources corresponding to the same target resource, can be coalesced into the same monitor object at the Proxy -- possibly with the same URI. In such case, a set containing one entry for each registered client is kept, which holds the client identities, their expiry and one or more preferred media types for their representation(s). When all entries are deleted (either because clients have explicitly deregistered the monitor, or the monitor period has expired), the corresponding "monitor" object is

deleted. Note that an underlying cache entry MAY still be kept in case the cached representation(s) are still fresh (i.e. the Max-Age of the "monitor" resource and Max-Age of the target resource have completely different semantics.)

If the monitor resource is successfully created, the server MUST return a 2.01 response containing one or more Location-Path and/or Location-Query Options to identify the monitored resource instance, which can be used from now on by the requester as an alias to the target resource.

At a later time, the client wakes up and wants to access the monitored resource. It does so by requesting the Proxy monitor resource that has been previously created.

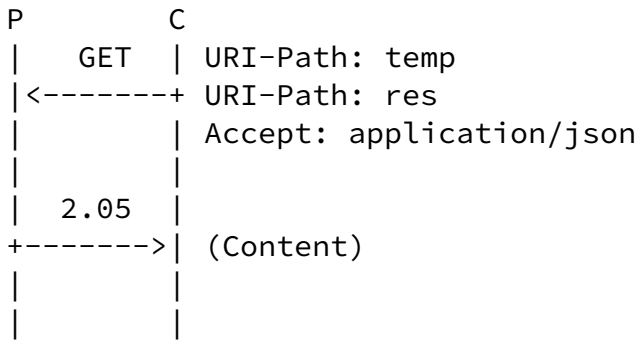


Figure 4

In case the observation on the target node has not been started because the Proxy has not yet been able to contact the origin, the Proxy will return a [TBD] error code.

In case the requested resource was not present on the origin, the Proxy will return an empty response (i.e. one with no payload.)

[[XXX: add an explicit response code perhaps like HTTP 204 ?]]

In case the monitor resource is not found in the Proxy, either because the Proxy has rebooted and lost its state, or the monitor resource has been de-registered (see Section 2.2.2), a 4.04 response code is returned to the client -- that can recreate it, if needed.

2.2.2. Monitor De-registration

The monitor object MUST be deleted at the Proxy when all its associated resources have been de-registered or have expired.

In order to save storage, a Proxy MAY decide to delete a monitor resource in case it has not been requested for a sufficiently long

time, or for any other reason. Note that the Proxy may also reboot and lose its state, including the state associated to any monitored resource. The requester can realize that the state at the Proxy has been lost, and re-instantiate the monitor, when it receives an unexpected 4.04 from the "monitor" resource.

2.2.2.1. Explicit De-registration

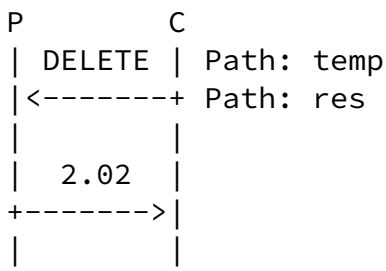


Figure 5

Explicit de-registration is performed by a client, with a DELETE on the URI returned by the Proxy on the corresponding registration.

2.2.2.2. Implicit De-registration

Implicit de-registration MUST occur when the monitoring period specified by the client via Max-Age expires. If no Max-Age was supplied at registration time, a default of 3600 seconds MUST be assumed.

2.2.3. Resource Refresh

In order to minimize the number of messages used by the monitoring process, the Proxy MUST try to install an observation on the requested resource. In case this first attempt fails, the Proxy MAY fall back to repeated poll whose duration is upper bounded by the Max-Age value indicated by the client during registration.

Usual cache validation MUST be applied to the cached copy of the monitored resource.

3. Acknowledgements

Bruce Nordman.

4. IANA Considerations

The following entries are added to the CoAP Option Numbers registry:

Number	Name	Reference
2n+1	Publish	RFC XXXX
2m+1	Monitor	RFC XXXX

5. Security Considerations

Threat: cache poisoning.
Countermeasure: authenticate sender.

Threat: unauthorized de-registration
Countermeasure: authenticate requester.

Threat: Proxy resources' exhaustion.
Countermeasure: authenticate requester + quota limit.

Threat: global state loss.
Countermeasure: cache redundancy.

6. References

6.1. Normative References

[I-D.ietf-core-coap]
Frank, B., Bormann, C., Hartke, K., and Z. Shelby, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-08 (work in progress), October 2011.

[I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format", draft-ietf-core-link-format-11 (work in progress), January 2012.

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-04 (work in progress), February 2012.

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

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

RFC 3986, January 2005.

6.2. 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.

[I-D.shelby-core-resource-directory]
Krco, S. and Z. Shelby, "CoRE Resource Directory", draft-shelby-core-resource-directory-02 (work in progress), October 2011.

Authors' Addresses

Thomas Fossati
KoanLogic
Via di Sabbiuno, 11/5
Bologna 40100
Italy

Email: tho@koanlogic.com

Pierpaolo Giacomin
Freelance

Email: yrz@anche.no

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com