Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2013

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

July 9, 2012

Monitor Option for CoAP
draft-fossati-core-monitor-option-00

Abstract

This memo defines Monitor, an additional Option for the Constrained Application Protocol (CoAP) especially targeted at sleepy sensors.

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 January 10, 2013.

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 on 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 an extension 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 requirement REQ3 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. Monitor Option

No.	C/E	Name	Format	Length	Default
XX	Critical	Monitor	(none)	0 B	(none)

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

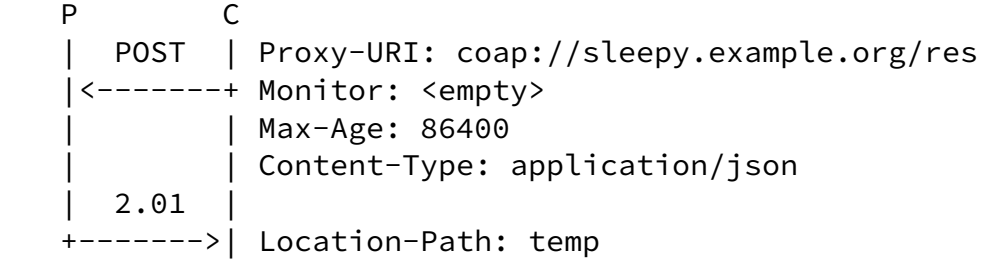
involved.

Suppose that the resource of interest is not cached anywhere, and a sleepy endpoint wants to Observe it through a 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.

2.1. Public Monitor Registration



```
|        | Location-Path: res
|        |
```

Figure 1

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. 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.

```
P        C
|   GET  | URI-Path: temp
```

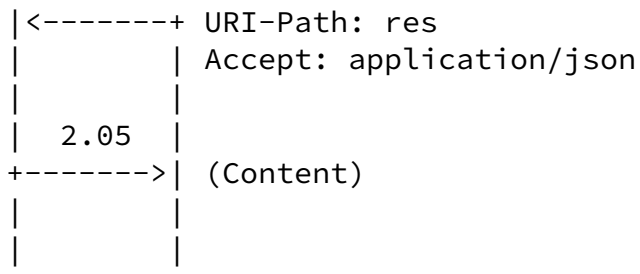


Figure 2

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), a 4.04 response code is returned to the client -- that can recreate it, if needed.

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.

2.2.1. Explicit De-registration

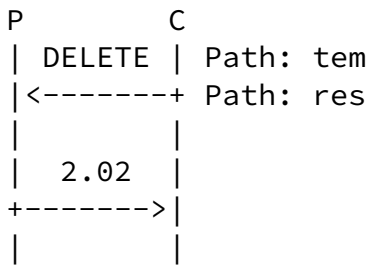


Figure 3

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. 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.

2.3. 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 and Matthieu Vial for discussing and giving advice on some of the ideas contained in this document.

4. IANA Considerations

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

Number	Name	Reference
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.

Threat: DoS on remote constrained resource via unneeded monitoring.
Countermeasure: access control on the constrained resource (?)

6. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-10 (work in progress), June 2012.

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-05 (work in progress), March 2012.

[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-04 (work in progress), May 2011.

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

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