Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 10, 2014

T. Fossati
Alcatel-Lucent
P. Giacomin
Freelance
S. Loreto
Ericsson

January 06, 2014

Publish Option for CoAP
draft-fossati-core-publish-option-03

Abstract

This memo defines the Publish Option for the Constrained Application Protocol (CoAP). The Publish Option is used by a CoAP Endpoint to control the authority delegation of one of its resources to another Endpoint. All the phases of the authority delegation process (setup, renewal, cancellation) are controlled by a simple RESTful protocol.

This memo also introduces the 'proxies' Web Linking relation type, to be used by a CoAP Proxy to explicitly advertise the resources that it can serve - either from its cache, or by forwarding the Client's request upstream.

The Publish Option and the 'proxies' relation provide the building blocks for a comprehensive, in-protocol, solution to the sleepy/ intermittent Endpoint use case.

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 July 10, 2014.

Copyright Notice

Table of Contents

1. Introduction

This memo defines the Publish Option for the Constrained Application Protocol [I-D.ietf-core-coap]. The Publish Option is used by a sleepy Endpoint (SEP) to temporarily delegate the authority of one of its resources to another, always on, Endpoint. The delegated Endpoint is typically a Proxy, though it could be an Endpoint with no other special network role. The SEP is given a simple RESTful messaging protocol that enables the setup, renewal and cancellation of the authority transfer. The whole process is driven by the SEP, which may actually never need to listen or to keep any state.

This memo also introduces the 'proxies' Web Linking [RFC5988] relation type. This new relation, which complements the default 'hosts' relation defined in [RFC6690], can be used by a CoAP Proxy to explicitly advertise the resources that it can serve, either from cache or by forwarding the Client's request upstream.

The 'proxies' relation works in concert with the Publish Option to enable SEP discovery even while SEP is off-line.

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].

The terms Client, Proxy, Server, and Endpoint are to be interpreted as described in [I-D.ietf-core-coap].

This memo reuses the terminology introduced in [I-D.rahman-core-sleepy-problem-statement], and aims at meeting the objectives stated in its Section 4 via an entirely in-protocol solution.

2. Publish Option

The Publish Option enables a SEP to temporarily (i.e. for a specified "lease" time) delegate the authority of one of its hosted resources to another Endpoint.

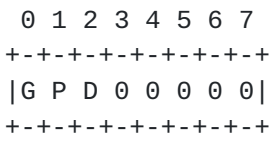
No.	C	U	N	R	Name	Format	Length	Default
31	x	x	x	-	Publish	uint	1	(none)

The one-byte integer value carried by the Publish Option allows the publishing node to specify the set of CoAP methods that are allowed on the resource (see Section 2.1 for details).

The "lease" time of the Publish action is specified by an associated (implicit or explicit) Max-Age Option value.

2.1. Value Format

The Publish Option consists of a single byte having the following layout:



Each of the higher 3 bits is a flag field indicating whether the associated CoAP method (respectively: GET, PUT and DELETE) is allowed on the published resource. The POST method has resource/application specific semantics and can't therefore be safely delegated. The lower 5 bits are reserved and MUST be set to 0.

The 0x00 value is used to explicitly revoke the delegation (see Section 2.2.3.) and MUST NOT be used for any other purpose of the Option.

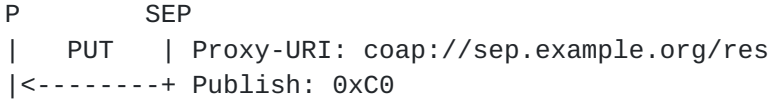
If the delegated Proxy receives a request for the published resource with a method that is not compatible with the mask supplied by the SEP, it MUST respond with a 4.05 (Method Not Allowed) response code.

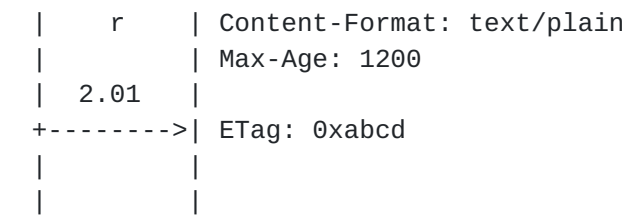
2.2. Operations

2.2.1. Publishing a Resource

The SEP publishes one of its hosted resources, specified by the enclosed Proxy-URI, by making a PUT 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).

The example below shows a delegation where the GET and PUT methods are allowed, whereas DELETE is explicitly prohibited, meaning that a Client can only read and update the resource.





The Proxy, which is voluntarily entrusted by the resource owner to act as the delegated origin for the "lease" time specified by Max-Age, replies with a 2.01 (Created) if the authority transfer succeeds. An exact duplicate of the submitted representation is created, and from now on it can be accessed via the delegated Proxy using the original URI encoded in a Proxy-Uri Option. If the Publish operation isn't successful (e.g. because the Proxy does not support Publish), then the origin transfer fails, and an appropriate response code is returned (e.g. 4.02 Bad Option).

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 Max-Age is elapsed, the Proxy MUST delete the published resource value (and any associated link-format metadata) and fall back to its usual proxying function.

On successful delegation, the Proxy MUST generate a new ETag and return it in the 2.01 response to the Client; if the published resource can be UPDATE'd, then the Client SHOULD save the ETag value (see Section 2.2.4).

The returned ETag value represents the state of the resource at the time the Publish operation is performed. The Proxy MUST change its value whenever the underlying resource representation changes, e.g. if it gets UPDATE'd. The current ETag value SHOULD be included by the Proxy in all responses involving the published representation. The ETag can be used by SEP to make conditional requests to the Proxy to check whether the representation has changed (see Section 2.2.4 for details).

The Publish Option is critical, and MUST NOT be present in a response. 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), a 4.00 (Bad Request) MUST be returned to the Client. The Publish Option is not Safe-to-Forward, and neither is a Cache-Key.

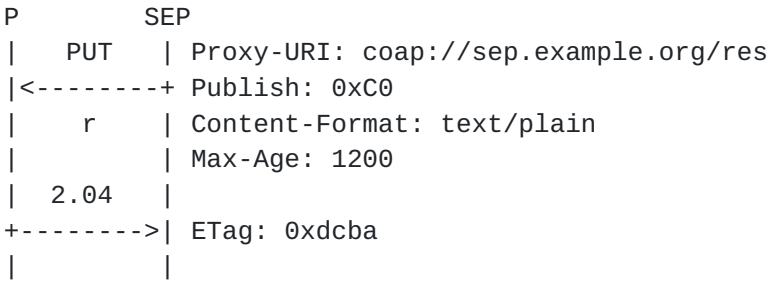
Since the 2.01 is emitted, and for the duration of the delegation, any Client wishing to access the resource can do so by making a Proxy-URI request to the Proxy, which shall then serve the resource from its own storage.

An interesting outcome of this communication strategy is that the SEP may really never need to listen on its radio interface. However, ignoring the response status code from Proxy, as well as the ETag value in case of UPDATE-able resources, is not a safe practice and SHOULD not be used unless the consequences are fully understood.

Upon publishing, the Proxy MUST save the identity (e.g. the IP address) of the publishing SEP, and MUST use it to correctly authorise "maintenance" operations such as renewal or cancellation of the published resource. The SEP identity MUST be kept for the whole duration of the delegation (including any associated renewal) and can be forgotten as soon as the delegation vanishes, either implicitly or explicitly.

2.2.2. Updating a Resource

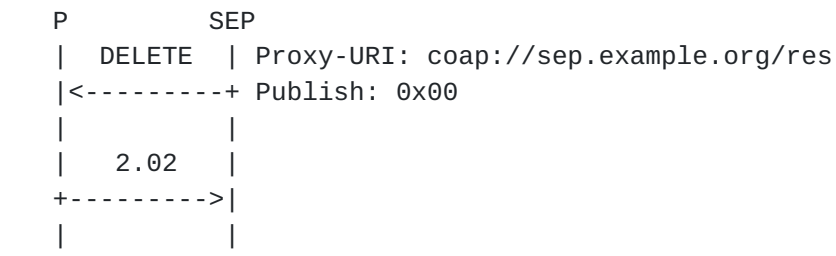
In order to update the delegated resource state or to just extend the lease period, the SEP sends basically the same request (except for the possibly updated representation value) to the Proxy, which in turn replies with a 2.04 Changed status code, and a new ETag value, in case the update operation succeeds. If the operation fails, e.g. because the request comes from an Endpoint different from the publishing SEP, a suitable status code is returned (e.g. 4.01 Unauthorized).



2.2.3. Unpublishing a Resource

The delegation of a given resource can be explicitly revoked by the SEP 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 0x00.

On successful deletion of the delegation, a 2.02 Deleted response code is returned by the Proxy. On error a suitable status code is returned.



2.2.4. Checking for Change

In order to check whether an UPDATE-able resource has changed, SEP issues a GET for the published resource with If-Match Option set to the last seen ETag value.

The possible outcomes are:

- 4.04 (Not Found) if the resource has been deleted;
- 2.05 (Content) if it has been otherwise modified;
- 2.03 (Valid) if it has not changed.

In case a 2.05 is returned, SEP saves the updated ETag returned by the Proxy, and uses it on subsequent If-Match GET's.

Note that, in exceptionally simple scenarios, an unconditional GET followed by a memcmp against the previous representation value, MAY constitute a viable alternative to the method described above.

3. The 'proxies' Relation Type

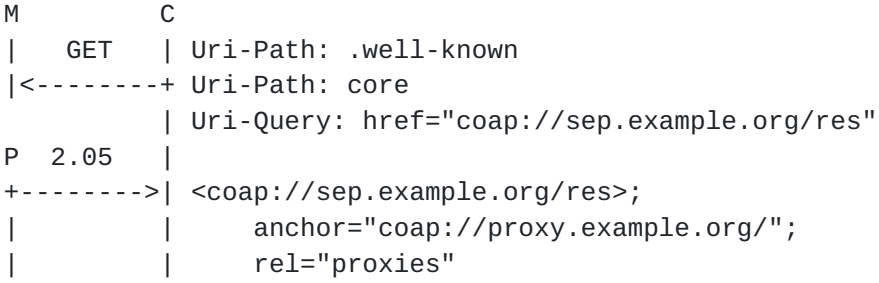
The new 'proxies' Web Linking [RFC5988] relation type is meant to signify that the target resource carried by the link, which MUST be identified by an absolute URI, is reachable through a Proxy-URI request made to the anchored Origin (i.e. the Proxy).

(Note that we need to specify the Proxy through an explicit anchor, thus increasing the verbosity of the link value, because of the way the context URI override rules are defined in Section 2.1 of [RFC6690]. In fact, absent an explicit anchor, rule (b) would set the context to the SEP origin, which is definitely not what we want.)

3.1. Examples

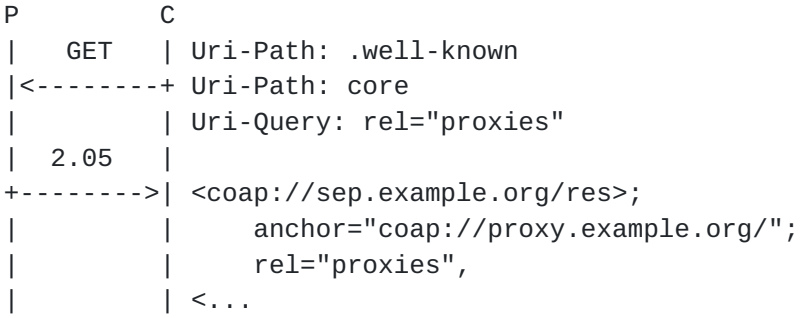
3.1.1. Discover the Proxy for a Resource

C multicasts a query to the /.well-known/core interface and discovers the P (associated to the coap://proxy.example.org authority) "proxies" the resource queried via an explicit href:

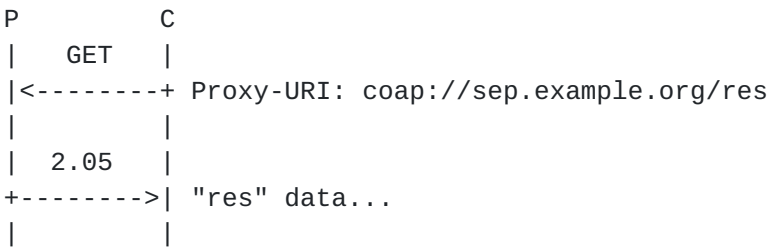


3.1.2. Discover all the Resources that an Endpoint 'proxies'

C discovers all the resources that P "proxies":



and can then GET one of the "proxied" resource from P:



The 'proxies' relation is orthogonal to the Publish Option, so it's up to P to decide whether to serve coap://sep.example.org/res from its store/cache, or to forward the request to the origin at coap://sep.example.org.

3.2. Publish Link-Format Attributes

3.2.1. Implicitly

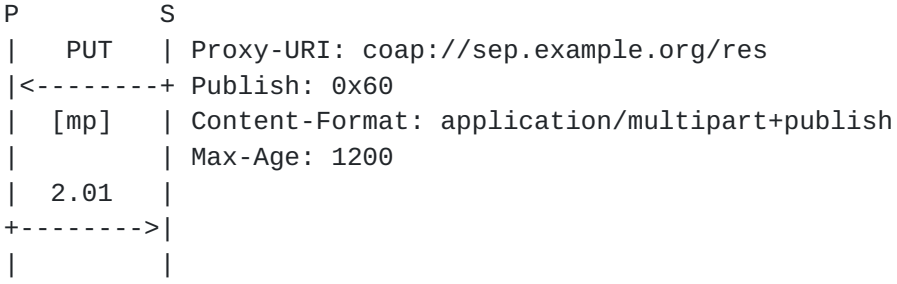
The resource metadata are implicitly extracted from the published representation. Basically, the Proxy works out the 'ct' and 'sz' attributes by inspecting Content-Format and the request payload size.

The main advantage of this method is that it needs no further transmission except that needed for the Publish operation. The disadvantage is the very limited (and fixed) number of attributes that can be derived, which makes it suitable only for the most basic use cases.

3.2.2. Explicitly

The resource metadata are explicitly published to the same Proxy-URI used for the sibling resource, either in a separate request/response cycle:

or atomically, within the same Publish operation, e.g. by using the Multipart Content-Format to aggregate one (or even more than one) representation(s) together with the application/link-format entry:



Note that the former is non-atomic, and limited to only one representation of the resource; the latter is atomic and supports multiple Content-Format's for the published resource.

4. Acknowledgements

Thanks to Bruce Nordman, Matthieu Vial, Akbar Rahman, and Esko Dijk for comments and discussions that have helped shaping this document.

5. IANA Considerations

The following entry is added to the CoAP Option Numbers registry:

Number	Name	Reference
31	Publish	This memo

This memo registers the new "proxies" Web Linking relation type as per [RFC5988].

Relation Name: proxies
Description: the target is the absolute URI of a resource proxied by the Origin stated in the anchor.
Reference: this memo
Notes: This relation is used in CoRE where links are retrieved as a "/.well-known/core" resource representation.
Application Data: None

6. Security Considerations

This section identifies Threats (T) and related countermeasures (C).

- T: cache poisoning.
- C: use strong auth to identify SEP.

- T: unauthorized update or de-registration
- C: strong auth to identify SEP.

- T: Proxy resources' exhaustion.
- C: use strong auth to identify SEP + quota limit.

- T: Inject fake copies of the resource by a 3rd party.
- C: use delegation scheme that bundles the identities of the SEP and the Proxy, together with the resource being delegated. A third party must be able to verify SEP and Proxy identities, maybe offline, and check the resource fingerprint.

6.1. Securing the Delegation

[[The following is just a sketch which needs further elaboration]] SEP signs the identity of the delegated Proxy and a fingerprint of the resource (both data and meta), and bundles it up with the resource itself, maybe in a MultiPart envelope (TBD define signed Content-Format). Client verifies the resource is indeed from the SEP by checking the signature, and it has been served by the intended origin, within the validity frame of the delegation. There seems to be an issue with hierarchical caching: the resource can't be served from a downstream Proxy which is different from the one that was originally delegated unless each Proxy in the delivery chain wraps the received message with its own credentials?

7. References

7.1. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

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

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

7.2. Informative References

[I-D.rahman-core-sleepy-problem-statement]
Rahman, A., Fossati, T., Loreto, S., and M. Vial, "Sleepy Devices in CoAP - Problem Statement", draft-rahman-core-sleepy-problem-statement-01 (work in progress), October 2012.

Appendix A. A (fairly) Comprehensive Example

The following section details the whole life-cycle of an hypothetical Sleepy/Intermittent node that uses Publish to exchange data (both reading and writing) with other agents in a CoAP network.

A.1. Actors

SEP Sleeping/Intermittent endpoint implementing two functions: F1, and F2. Each function exposes one configurable parameter, and provides one output.

P Proxy with Publish support.

W Controller application which can configure function parameters on SEP.

R Consumer application which reads values from SEP.

A.2. Resources

The following resources model the two functions (F1 and F2) implemented by SEP in terms of their input and output parameters:

coap://sep1/i1 Configurable parameter for F1.

coap://sep1/i2 Configurable parameter for F2.

coap://sep1/o1 Output of F1.

coap://sep1/o2 Output of F2.

If the number of configuration parameters is not trivially small, then it might be handy to create an aux resource which can be polled by the SEP to track the parameters that have been reconfigured:

coap://sep1/im Update parameter mask. Conceptually a n-bit mask (one bit per configurable parameter) used by W to mark the updated parameters, and by SEP to clear them once the corresponding configuration has been applied.

A.3. Application Flow

A.3.1. Bootstrap

SEP publishes all the application resources to P.

Configurable parameter for F1:

```
SEP -> P : PUT Proxy-URI=coap://sep1/i1, Publish="G,U", Payload="1"
P -> SEP : 2.01 (Created), ETag=0x01
```

Configurable parameter for F2:

```
SEP -> P : PUT Proxy-URI=coap://sep1/i2, Publish="G,U", Payload="2"
P -> SEP : 2.01 (Created), ETag=0x01

Output of F1:

SEP -> P : PUT Proxy-URI=coap://sep1/o1, Publish="G", Payload=""
P -> SEP : 2.01 (Created), ETag=0x01

Output of F2:

SEP -> P : PUT Proxy-URI=coap://sep1/o2, Publish="G", Payload=""
P -> SEP : 2.01 (Created), ETag=0x01
```

This assumes that SEP has pre-canned values "1" and "2" for its configurable parameters i1 and i2 respectively.

```
Optionally:

SEP -> P : PUT Proxy-URI=coap://sep1/im, Publish="G,U",
           Payload="update_mask_cleared"
P -> SEP : 2.01 (Created), ETag=0x01
```

A.3.2. Configuration and Reconfiguration

```
W sets a new value, e.g. 5, for i2:

W -> P :   PUT Proxy-URI=coap://sep1/i2, Payload="5"
P -> W :   2.04 (Changed), ETag=0x02
```

P updates the value of i2 accordingly, and sets a new ETag on it, e.g. 0x02.

When SEP wakes up, it polls its configuration variables via a conditional GET that uses the ETags returned by P at publishing time. Since i1 has not changed, and is still associated with the original ETag, a 2.03 status code is returned:

```
SEP -> P : GET Proxy-URI=coap://sep1/i1, If-Match=0x01
P -> SEP : 2.03 (Valid)
```

Since i2 has changed, a 2.05 status code is returned and the payload carries the new value. Also, the new ETag associated with i2 is returned and is updated locally by the SEP:

```
SEP -> P : GET Proxy-URI=coap://sep1/i2, If-Match=0x01
P -> SEP : 2.05 (Content), ETag=0x02, Payload="5"
```

The SEP reconfigures its F1 based on the new configuration setting, and continues its operations.

A.3.3. Updating Functional Output

SEP wakes up and commits the newly computed values, e.g. 6 and 8, to P:

```
SEP -> P : PUT Proxy-URI=coap://sep1/o1, Publish="G", Payload="6"
P -> SEP : 2.04 (Changed), ETag=0x02

SEP -> P : PUT Proxy-URI=coap://sep1/o2, Publish="G", Payload="8"
P -> SEP : 2.04 (Changed), ETag=0x02
```

P sets the new values, assigns a new ETag, and gives it back to P together with a 2.04 status code.

A.3.4. Retrieving Functional Output

R needs to retrieve the latest values for the functions computed by SEP; thus, it asks P to retrieve the associated resources:

```
R -> P :   GET Proxy-URI=coap://sep1/o1
P -> R :   2.05 (Content), ETag=0x02, Payload="6"

R -> P :   GET Proxy-URI=coap://sep1/o2
P -> R :   2.05 (Content), ETag=0x02, Payload="8"
```

Note that the exchange above applies to the very first poll. Subsequent polls can be done conditionally on the "last-seen" ETag.

Also note that the above assumes SEP has been able to update its values at least once. R must be prepared to retrieve empty representations, if SEP has not yet updated their value since bootstrap.

A.3.5. SEP Reboot

The idempotence of all the involved methods guarantees a clean recovery in face of a reboot of the SEP. In fact, if at a given time SEP reboots and loose soft state, including the configuration parameters: SEP has to go again through the bootstrap phase in which the application resources are published:

```
SEP -> P : PUT Proxy-URI=coap://sep1/i1, Publish="G,U", Payload="1"
P -> SEP : 2.04 (Changed), ETag=0x03

SEP -> P : PUT Proxy-URI=coap://sep1/i2, Publish="G,U", Payload="2"
```

```
   P -> SEP : 2.04 (Changed), ETag=0x03

   SEP -> P : PUT Proxy-URI=coap://sep1/o1, Publish="G", Payload=""
   P -> SEP : 2.04 (Changed), ETag=0x03

   SEP -> P : PUT Proxy-URI=coap://sep1/o2, Publish="G", Payload=""
   P -> SEP : 2.04 (Changed), ETag=0x03
```

The ETag's value (0x03) needs to be not recently used (not within the Max-Age period for the resource).

From now on everything can proceed as described in Appendix A.3.2, Appendix A.3.3, and Appendix A.3.4

Authors' Addresses

Thomas Fossati
Alcatel-Lucent
3 Ely Road
Milton, Cambridge CB24 6DD
UK

Email: thomas.fossati@alcatel-lucent.com

Pierpaolo Giacomin
Freelance

Email: yrz@anche.no

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com