

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.

Internet-Draft

CoAP-observe

October 2010

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

Internet-Draft

CoAP-observe

October 2010

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

Client	Server
CON tid=47	
GET /foo	
lifetime=60s	
+----->	

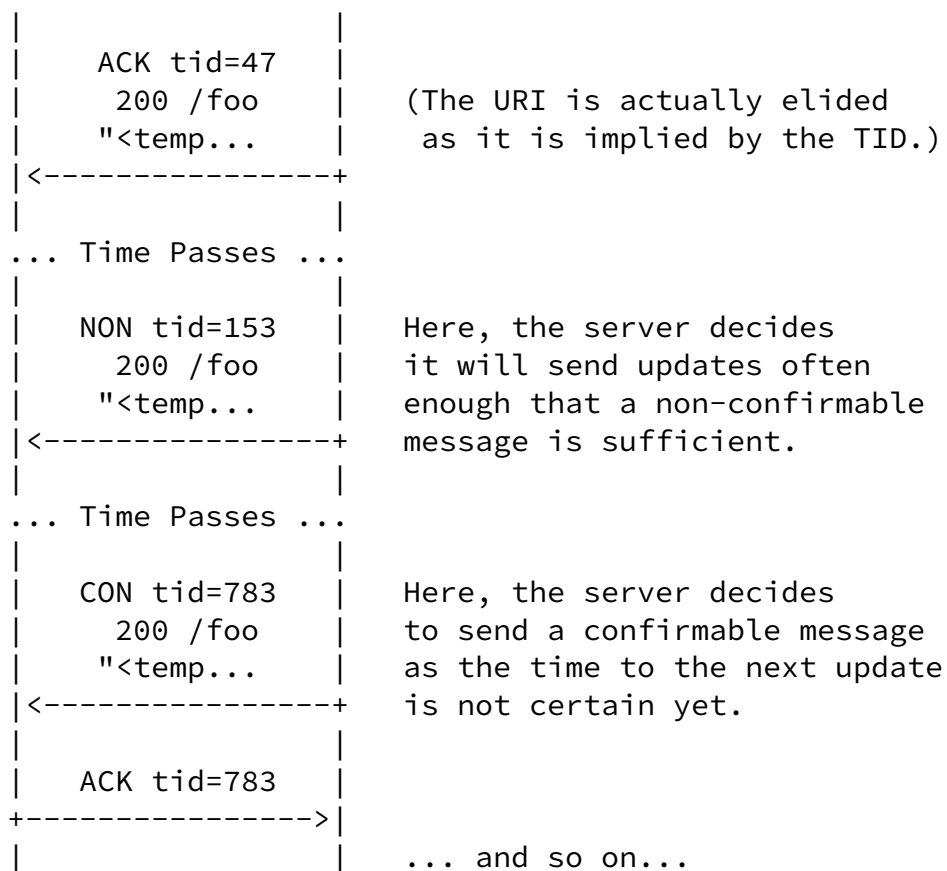


Figure 1

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.

Internet-Draft

CoAP-observe

October 2010

[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

Internet-Draft

CoAP-observe

October 2010

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

