Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 4, 2015

M. Koster
ARM Limited
A. Keranen
J. Jimenez
Ericsson

March 3, 2015

Publish-Subscribe in the Constrained Application Protocol (CoAP)
draft-koster-core-coap-pubsub-01

Abstract

The Constrained Application Protocol, CoAP, and related extensions are intended to support machine-to-machine communication in systems where one or more nodes are resource constrained, in particular for low power wireless sensor networks. This document defines publish-subscribe and message queuing functionality for CoAP that extends the capabilities for supporting nodes with long breaks in connectivity and/or up-time.

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 4, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] supports machine to machine communication across networks of constrained devices. One important class of constrained devices includes devices that are intended to run for years from a small battery, or by scavenging energy from their environment. These devices spend most of their time in a sleeping state with no network connectivity.

Devices may also have limited reachability due to certain middle-boxes, such as Network Address Translators (NATs) or firewalls. Such devices must communicate using a client role, whereby the endpoint is responsible for initiating communication.

This document specifies the means for nodes with limited reachability to communicate using simple extensions to CoAP and the CoRE Resource Directory [I-D.ietf-core-resource-directory]. The extensions enable publish-subscribe communication using a broker node that enables store-and-forward messaging between two or more nodes.

The mechanisms specified in this document are meant to address key design requirements from earlier CoRE drafts covering sleepy node support and mirror server.

2. Terminology

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this specification are to be interpreted as described in [RFC2119].

This specification requires readers to be familiar with all the terms and concepts that are discussed in [RFC5988] and [RFC6690]. Readers should also be familiar with the terms and concepts discussed in [RFC7252] and [I-D.ietf-core-resource-directory]. The URI template format, see [RFC6570], is used to describe the REST interfaces defined in this specification.

This specification makes use of the following additional terminology:

CoAP Publish-Subscribe (CoAP-PubSub) Service: A service provided by a node or system where CoAP messages sent by one endpoint to another are queued (stored) by intermediate node(s) and forwarded only when suitable, e.g., when the message recipient endpoint is not sleeping.

CoAP-PubSub Broker: A server node capable of storing messages to and from other nodes and able to match subscriptions and publications in order to route messages to right destinations.

CoAP-PubSub function set: A group of well-known REST resources that together provide the CoAP-PubSub service.

CoAP-PubSub Client A client that implements the CoAP-PubSub function set.

Publish-Subscribe (pub-sub): A messaging paradigm where messages are published (e.g., to a broker) and potential receivers can subscribe to receive the messages.

Topic: In Publish-Subscribe systems a topic is a unique identifying string for a particular item or object being published and/or subscribed to.

3. Architecture

3.1. CoAP-PubSub Architecture

Figure 1 Figure 1 shows the architecture of a CoAP PubSub service. CoAP PubSub Clients interact with a CoAP PubSub Broker through the CoAP PubSub interface which is hosted by the Broker. State information is updated between the Clients and the Broker. The CoAP PubSub Broker performs a store-and-forward function of state updates between certain CoAP PubSub Clients. Clients Subscribe to state updates which are Published by other Clients, and which are forwarded by the Broker to the subscribing clients. The CoAP PubSub Broker also acts as a REST proxy, retaining the last state update provided by clients to supply in response to Read requests from Clients.

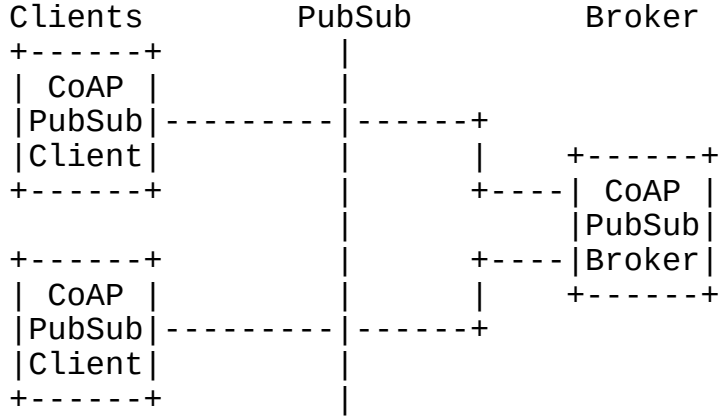

Figure 1: CoAP-PubSub Architecture

3.2. CoAP PubSub Broker

A CoAP PubSub Broker is a CoAP Server that exposes an interface for clients to use to initiate publish-subscribe interactions.

3.3. CoAP PubSub Client

A CoAP PubSub Client interacts with a CoAP PubSub Broker using the CoAP PubSub interface. Clients initiate all interactions with the CoAP-PubSub broker. Sensor Clients Publish state updates to the Broker. Actuator Clients read from or subscribe to state updates from the broker. Application clients make use of both publish and subscribe in order to exchange state upadates with Sensors and Actuators.

3.4. CoAP PubSub Topic

A PubSub Topic is a strings used to identify particular resources and objects in a publish-subscribe system. Topics are conventionally formed as a hierarchy, e.g. "/sensors/weather/barometer/pressure" or "EP-33543/sen/3303/0/5700".

4. CoAP PubSub Function Set

This section is normative.

This section defines the interfaces between a CoAP PubSub Broker and PubSub Clients, which is called the CoAP PubSub Function Set. The examples throughout this section assume the use of CoAP [RFC7252]. A CoAP PubSub Broker implementing this specification MUST support the discover, create, publish, subscribe, and remove operations defined in this section.

4.1. DISCOVER

CoAP PubSub Clients discover CoAP PubSub Brokers by uisng CoAP Simple Discovery or through Resource Directory. A CoAP PubSub Broker MAY indicate its presence and availability on a network by exposing a link to its PubSub function set at its .well-known/core location. A CoAP PubSub broker MAY register its PubSub function set location with a Resource Directory.

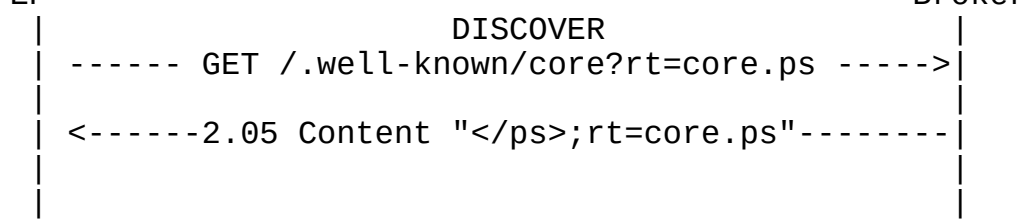

Figure 2: DISCOVER

4.2. CREATE

Clients MAY Create Topics on the broker. A client wishing to cerate a topic MUST use CoAP POST to the PubSub function set location with a payload of the desired topic. The Topic MUST be a valid URI as described in [ref]. The client MAY indicate the lifetime of the topic by including the max-age option in the CREATE request. Broker

MUST return 2.xx if the topic is created and 4.xx if the topic can not be created. Broker MAY remove topics if max-age is excedded without any publishes to the topic. Broker MUST return 4.xx to clients attempting to Read or Subscribe to expired topics.

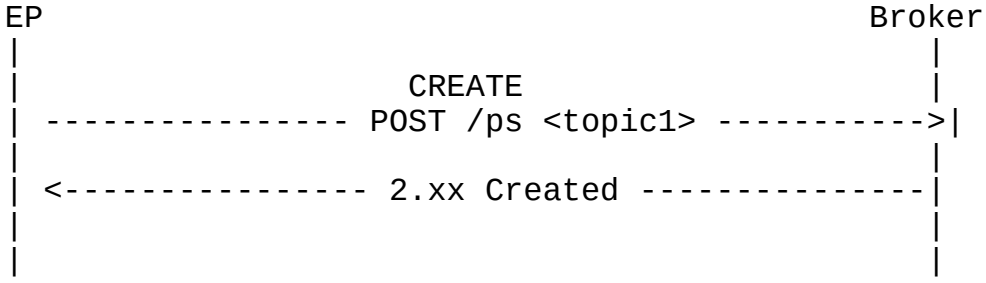

Figure 3: CREATE

4.3. PUBLISH

The Client endpoint PUBLISHes updates to CoAP-PubSub broker. A CoAP-PubSub client endpoint MUST use PUT to publish state updates to the CoAP-PubSub broker. A CoAP client publishing on a topic MAY indicate the maximum lifetime of the value by including the max-age option in the Publish request. The CoAp PubSub Broker MUST return 2.xx if the publish is accepted, or 4.xx if the topic does not exist.

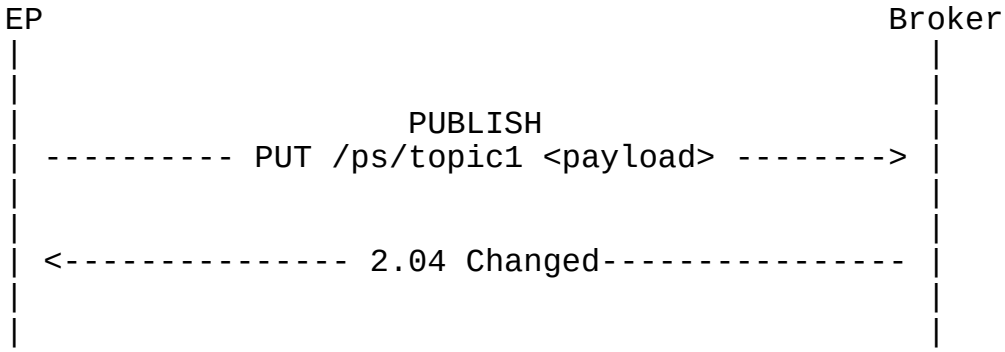

Figure 4: PUBLISH

4.4. SUBSCRIBE

CoAP PubSub Clients SUBSCRIBE to topics on the Broker using CoAP GET+Observe. A CoAP PubSub Client wishing to Subscribe to a topic on a Broker MUST use CoAP GET+Observe. The Broker MUST return a 2.05 Content response along with the most recent update if the topic

contains a valid value, or 4.xx if the value is expired, or 4.xx if the topic does not exist.

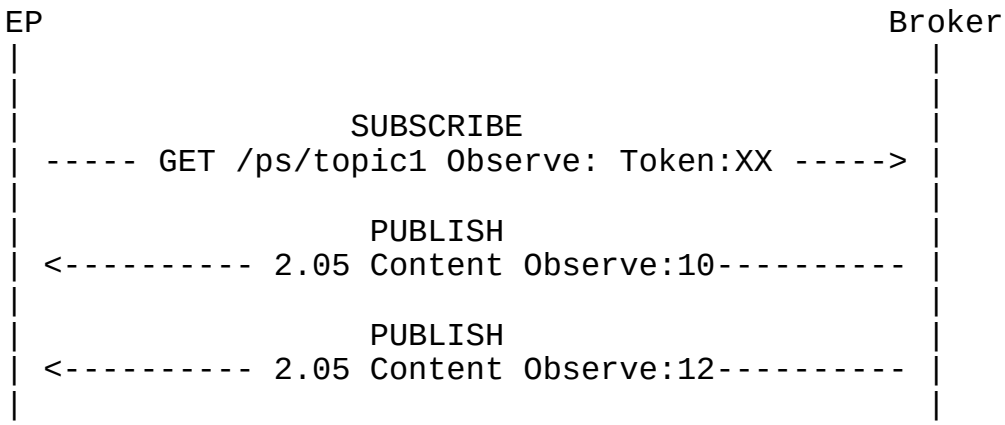

Figure 5: SUBSCRIBE

4.5. READ

A CoAP PubSub client wishing to obtain the most recent published value on a topic MAY Read the value from the broker. A client wishing to READ a topic from a broker MUST use the CoAP GET operation. he Broker MUST return a 2.05 Content response along with the most recent update if the topic contains a valid value, or 4.xx if the value is expired, or 4.xx if the topic does not exist.

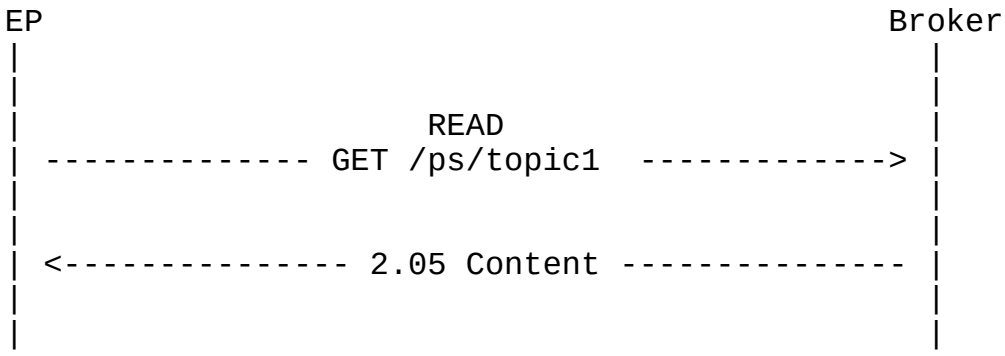

Figure 6: READ

4.6. REMOVE

A CoAp PubSub Client wishing to remove a topic MUST use a CoAP Delete operation on the URL of the topic. The CoAP PubSub Broker MUST return 2.xx Deleted if the Delete operation is successful, 4.xx if the operation is unsuccessful, and 4.xx if the topic does not exist.

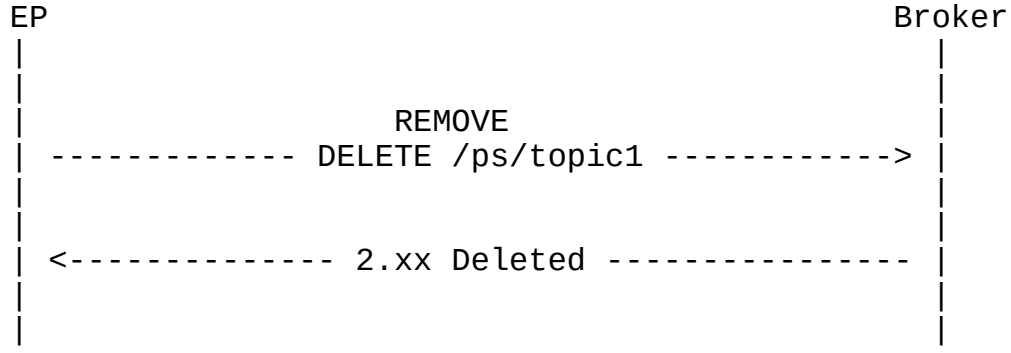

Figure 7: REMOVE

5. CoAP PubSub Operation with Resource Directory

A CoAP PubSub Broker may register a PubSub Function Set with a Resource Directory. A PubSub Client may use Resource Directory to discover a PubSub Broker.

A CoAP PubSub CLient may register CoRE Links (RFC6690) to created PubSub Topics with a Resource Directory. A PubSub Client may use Resource Directory to discover PubSub Topics. A client wishing to register PubSub Topics with a RD MUST use the context relation (CON) to indicate that the context of the registered links is the PubSub Broker.

6. Sleep-Wakeup Operation

A CoAP PubSub client may sleep between publishing state updates to the Broker. A CoAP PubSub client may sleep between retrieving state updates from the broker when using the READ operation to retrieve the state updates. A CoAP PubSub Client must not sleep while subscribed to state updates from the Broker.

7. Security Considerations

CoAP-PubSub re-uses CoAP [RFC7252], CoRE Resource Directory [I-D.ietf-core-resource-directory], and Web Linking [RFC5988] and therefore the security considerations of those documents also apply

to this specification. Additionally, a CoAP-PubSub broker and the endpoints SHOULD authenticate each other and enforce access control policies. A malicious EP could subscribe to data it is not authorized to or mount a denial of service attack against the broker by publishing a large number of resources. The authentication can be performed using the already standardized DTLS offered mechanisms, such as certificates. DTLS also allows communication security to be established to ensure integrity and confidentiality protection of the data exchanged between these relevant parties. Provisioning the necessary credentials, trust anchors and authorization policies is non-trivial and subject of ongoing work.

The use of a CoAP-PubSub broker introduces challenges for the use of end-to-end security between the end device and the cloud-based server infrastructure since brokers terminate the exchange. While running separate DTLS sessions from the EP to the broker and from broker to the web application protects confidentially on those paths, the client/server EP does not know whether the commands coming from the broker are actually coming from the client web application. Similarly, a client web application requesting data does not know whether the data originated on the server EP. For scenarios where end-to-end security is desirable the use of application layer security is unavoidable. Application layer security would then provide a guarantee to the client EP that any request originated at the client web application. Similarly, integrity protected sensor data from a server EP will also provide guarantee to the client web application that the data originated on the EP itself. The protected data can also be verified by the intermediate broker ensuring that it stores/caches correct request/response and no malicious messages/ requests are accepted. The broker would still be able to perform aggregation of data/requests collected.

Depending on the level of trust users and system designers place in the CoAP-PubSub broker, the use of end-to-end encryption may also be envisioned. The CoAP-PubSub broken would then only be able to verify the request/response message/commands and store-and-forward without being able to inspect the content. The solution for providing application layer security will depend on the utilized data encoding. For example, with a JSON-based data encoding the work from the JOSE working group could be re-used. Distribution of the credentials for accomplishing end-to-end security might introduce challenges if previously unknown parties need to exchange data.

8. IANA Considerations

This document registers two attribute values in the Resource Type (rt=) registry established with RFC 6690 [RFC6690].

8.1. Resource Type value 'core.pubsub.client'

- Attribute Value: core.pubsub.client
- Description: Section X of [[This document]]
- Reference: [[This document]]
- Notes: None

8.2. Resource Type value 'core.pubsub.server'

- Attribute Value: core.pubsub.server
- Description: Section Y of [[This document]]
- Reference: [[This document]]
- Notes: None

9. Acknowledgements

The authors would like to thank Hannes Tschofenig, Zach Shelby, Mohit Sethi, and Anders Eriksson for their contributions and reviews

10. References

10.1. Normative References

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-16 (work in progress), December 2014.

[I-D.ietf-core-resource-directory]
Shelby, Z. and C. Bormann, "CoRE Resource Directory", draft-ietf-core-resource-directory-02 (work in progress), November 2014.

[OMALightweightM2M]
Open Mobile Alliance, "OMA LightweightM2M v1.0", http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0, 12 2013.

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

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.

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

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

10.2. Informative References

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

Authors' Addresses

Michael Koster
ARM Limited

Email: Michael.Koster@arm.com

Ari Keranen
Ericsson

Email: ari.keranen@ericsson.com

Jaime Jimenez
Ericsson

Email: jaime.jimenez@ericsson.com