Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2015

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

March 7, 2015

Publish-Subscribe Broker for 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 a publish-subscribe broker for CoAP that extends the capabilities of CoAP 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 8, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] supports machine-to-machine communication across networks of constrained devices. CoAP uses a request/response model where clients make requests to servers in order to request actions on resources. Depending on the situation the same device may act either as a server or a client.

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 have limited reachability, because they 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 middle-boxes often prevent connecting to a device from the Internet unless the connection was initiated by the device.

This document specifies the means for nodes with limited reachability to communicate using simple extensions to CoAP. The extensions enable publish-subscribe communication using a broker node that enables store-and-forward messaging between two or more nodes. Furthermore the extensions facilitate many-to-many communication using CoAP.

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 [RFC6570] is used to describe the REST interfaces defined in this specification.

This specification makes use of the following additional terminology:

Publish-Subscribe (pub-sub): A messaging paradigm where messages are published to a broker and potential receivers can subscribe to the broker to receive messages. The publishers do not (need to) know where the message will be eventually sent: the publications and subscriptions are matched by a broker and publications are delivered by the broker to subscribed receivers.

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

CoAP-PubSub Broker: A server node capable of receiving messages (publications) to and from other nodes and able to match subscriptions and publications in order to route messages to right destinations. The broker can also temporarily store publications to satisfy future subscriptions.

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

Topic: A unique identifier for a particular item being published and/or subscribed to. A broker uses the topics to match subscriptions to publications.

3. Architecture

3.1. CoAP-PubSub Architecture

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.

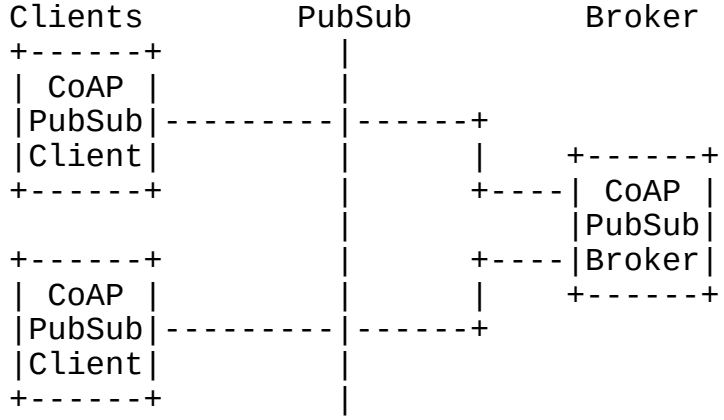

Figure 1: CoAP-PubSub Architecture

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 updates with Sensors and Actuators.

3.4. CoAP PubSub Topic

A PubSub Topic is a string used to identify a particular resource or object 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, UNSUBSCRIBE, READ, and REMOVE operations defined in this section. With the exception of PUBLISH, all operations in the CoAP PubSub Function Set MUST use confirmable (CON) CoAP messages.

4.1. DISCOVER

CoAP PubSub Clients discover CoAP PubSub Brokers by using 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 shows an example of a client discovering a local PubSub Function Set using CoAP Simple Discovery. A broker wishing to advertize the CoAP PubSub Function Set for Simple Discovery or through a Resource Directory MUST use the link relation rt="core.ps".

The DISCOVER interface is specified as follows:

Interaction: Client -> Broker

Method: GET

URI Template: /.well-known/core{?rt}

URI Template Variables:

rt := Resource Type (optional). MAY contain the value "core.ps"

Content-Format: application/link-format (if any)

The following response codes are defined for this interface:

Success: 2.05 "Content" with an application/link-format payload containing one or more matching entries for the broker resource. A PubSub broker SHOULD use the value "ps" for the link subject variable whenever possible.

Failure: 4.04 "Not Found" is returned in case no matching entry is found for a unicast request.

Failure: 4.00 "Bad Request" is returned in case of a malformed request for a unicast request.

Failure: No error response to a multicast request.

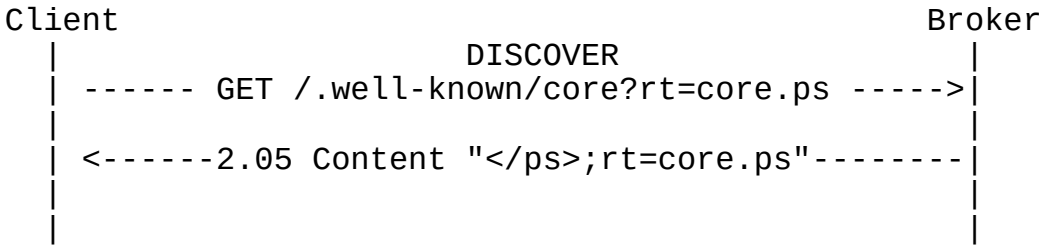

Figure 2: DISCOVER

4.2. CREATE

Clients MAY Create Topics on the broker. A client wishing to create a topic MUST use CoAP POST to the PubSub function set location with a payload indicating the desired topic. The Topic MUST be a valid URI as described in [RFC3986]. The client MAY indicate the lifetime of the topic by including the max-age option in the CREATE request. Broker MUST return a response code of 2.01 Created if the topic is created. The broker MUST return the appropriate 4.xx response code indicating the reason for failure if a new topic can not be created. Broker MAY remove topics if the max-age of the topic is exceeded without any publishes to the topic. Figure 3 shows an example of a topic called "topic1" being successfully created.

The CREATE interface is specified as follows:

Interaction: Client -> Broker

Method: POST

URI Template: /{+ps}

URI Template Variables:

ps := PubSub Function Set path (mandatory). The path of the PubSub Function Set, as obtained from discovery. A PubSub broker SHOULD use the value "ps" for this variable whenever possible.

Content-Format: text/plain

Payload: The desired topic to CREATE

The following response codes are defined for this interface:

Success: 2.01 "Created". Successful Creation of the topic

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.01 "Unauthorized". Authorization failure.

Failure: 4.03 "Forbidden". Topic already exists.

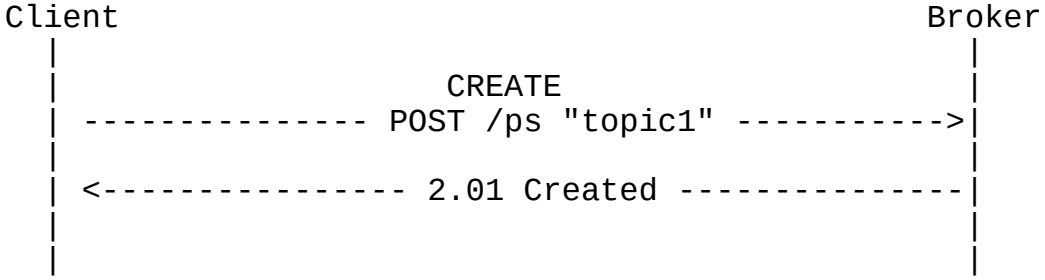

Figure 3: CREATE

4.3. PUBLISH

The CoAP PubSub Client PUBLISHes updates to CoAP-PubSub broker. A CoAP-PubSub client 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. A client MAY use confirmable (CON) or non-confirmable (NON) messages to PUBLISH updates to a broker. If the client PUBLISHes using a confirmable message, the Broker MUST return a response code of 2.04 Changed if the publish is accepted. If the client PUBLISHes using a confirmable message, the Broker MUST return a response code of 4.04 Not Found if the topic does not exist.

The Broker MUST PUBLISH updates to all clients subscribed on a particular topic each time it receives a PUBLISH on that topic. If a client PUBLISHes to a broker using non-confirmable messages, the broker MAY PUBLISH those messages to SUBSCRIBEd clients using non-confirmable messages. If a client PUBLISHes to a broker using confirmable messages, the broker MUST also PUBLISH those messages to SUBSCRIBEd clients using confirmable messages. If a client PUBLISHes to a broker with the max-age option, the broker MUST PUBLISH to SUBSCRIBEd clients including the same value for the max-age option. A broker MUST use CoAP Notification as described in [I-D.ietf-core-observe] to PUBLISH to subscribed clients.

The PUBLISH interface is specified as follows:

Interaction: Client -> Broker

Method: PUT

URI Template: /{+ps}/{topic}

URI Template Variables:

ps := PubSub Function Set path (mandatory). The path of the PubSub Function Set, as obtained from discovery.

topic := The desired topic to PUBLISH on.

Content-Format: Any valid CoAP Content Format

Payload: Representation of the topic value (CoAP resource state representation) in the indicated Content-Format

The following response codes are defined for this interface:

Success: 2.04 "Changed". Successful PUBLISH, topic is updated

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.01 "Unauthorized". Authorization failure.

Failure: 4.04 "Not Found". Topic does not exist.

Figure 4 shows an example of a new value being successfully published to the topic "topic1".

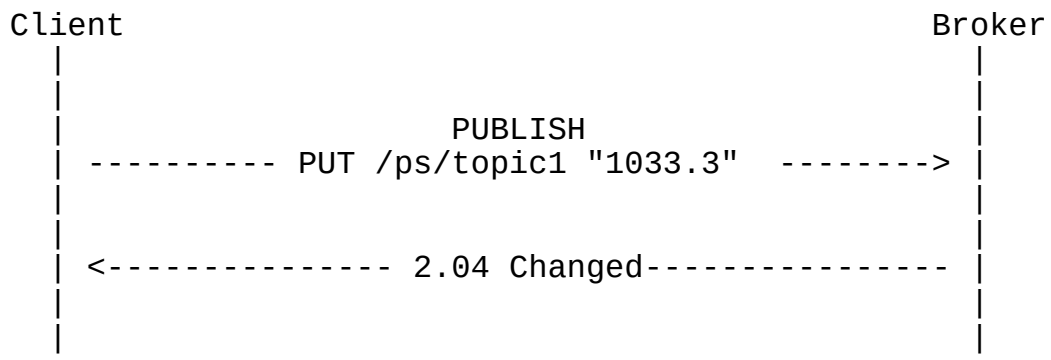

Figure 4: PUBLISH

4.4. SUBSCRIBE

CoAP PubSub Clients SUBSCRIBE to topics on the Broker using CoAP Observe as described in [I-D.ietf-core-observe]. A CoAP PubSub Client wishing to Subscribe to a topic on a Broker MUST use CoAP GET+Observe. The Broker MAY add the client to a list of observers. The Broker MUST return a response code of 2.05 Content along with the most recently published value if the topic contains a valid value. If the topic was PUBLISHed with the max-age option, the broker MUST set the max-age option in the valid response to the amount of time remaining for the topic to be valid since the last PUBLISH. The Broker MUST return a response code of 2.04 No Content if the max-age of the previously stored value has expired. The Broker MUST return a response code 4.04 Not Found if the topic does not exist or has been REMOVEd. Figure 5 shows an example of subscribing to "topic1" and receiving two PUBLISH responses from the broker.

The SUBSCRIBE interface is specified as follows:

Interaction: Client -> Broker

Method: GET+Observe

URI Template: /{+ps}/{topic}

URI Template Variables:

ps := PubSub Function Set path (mandatory). The path of the PubSub Function Set, as obtained from discovery.

topic := The desired topic to SUBSCRIBE to.

Content-Format: Any valid CoAP Content Format

Response Payload: Representation of the topic value (CoAP resource state representation) in the indicated Content-Format

The following response codes are defined for this interface:

Success: 2.05 "Content". Successful SUBSCRIBE, current value included

Success: 2.04 "No Content". Successful SUBSCRIBE, value not included

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.01 "Unauthorized". Authorization failure.

Failure: 4.04 "Not Found". Topic does not exist.

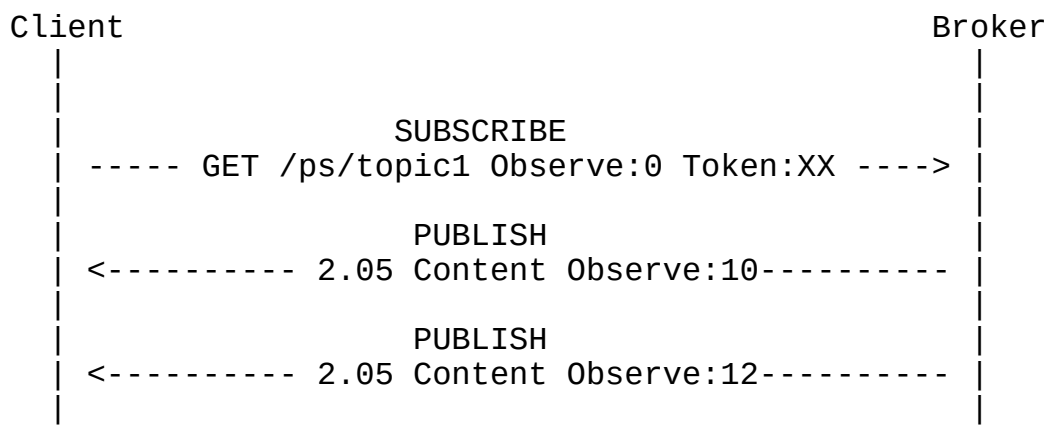

Figure 5: SUBSCRIBE

4.5. UNSUBSCRIBE

CoAP PubSub Clients UNSUBSCRIBE to topics on the Broker using the CoAP Cancel Observation operation. A CoAP PubSub Client wishing to UNSUBSCRIBE to a topic on a Broker MUST either use CoAP GET+Observe with an Observe parameter of 1 or send a CoAP Reset message in response to a PUBLISH [I-D.ietf-core-observe]. Figure 6 shows an example of a client UNSUBSCRIBE using the Observe=1 cancellation method.

The UNSUBSCRIBE interface is specified as follows:

Interaction: Client -> Broker

Method: GET+Observe

URI Template: /{+ps}/{topic}

URI Template Variables:

ps := PubSub Function Set path (mandatory). The path of the PubSub Function Set, as obtained from discovery.

topic := The desired topic to UNSUBSCRIBE from.

Content-Format: Any valid CoAP Content Format

Response Payload: Representation of the topic value (CoAP resource state representation) in the indicated Content-Format

The following response codes are defined for this interface:

Success: 2.05 "Content". Successful UNSUBSCRIBE, current value included

Success: 2.04 "No Content". Successful UNSUBSCRIBE, value not included

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.01 "Unauthorized". Authorization failure.

Failure: 4.04 "Not Found". Topic does not exist.

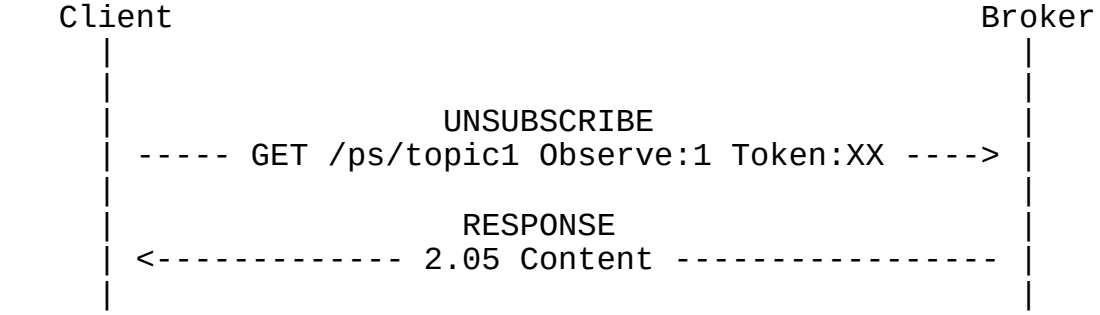

Figure 6: UNSUBSCRIBE

4.6. 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. The Broker MUST return a response code of 2.05 Content along with the most recently published value if the topic contains a valid value. If the topic was PUBLISHed with the max-age option, the broker MUST set the max-age option in the valid response to the amount of time remaining for the topic to be valid since the last PUBLISH. The Broker MUST return a response code of 2.04 No Content if the max-age of the previously stored value has expired. The Broker MUST return a response code 4.04 Not Found if the topic does not exist or has been REMOVEd. Figure 7 shows an example of s successful READ from topic1.

The READ interface is specified as follows:

Interaction: Client -> Broker

Method: GET

URI Template: /{+ps}/{topic}

URI Template Variables:

ps := PubSub Function Set path (mandatory). The path of the PubSub Function Set, as obtained from discovery.

topic := The desired topic to READ.

Content-Format: Any valid CoAP Content Format

Response Payload: Representation of the topic value (CoAP resource state representation) in the indicated Content-Format

The following response codes are defined for this interface:

Success: 2.05 "Content". Successful READ, current value included

Success: 2.04 "No Content". Topic exists, value not included

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.01 "Unauthorized". Authorization failure.

Failure: 4.04 "Not Found". Topic does not exist.

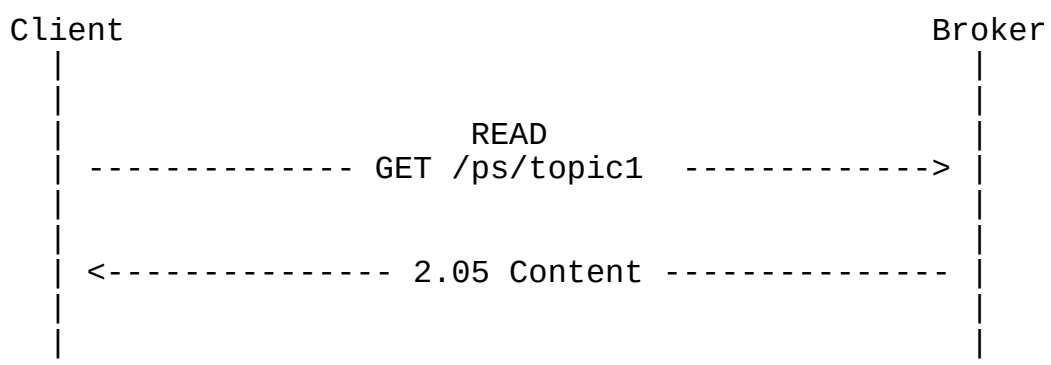

Figure 7: READ

4.7. REMOVE

A CoAP PubSub Client wishing to REMOVE a topic MUST use a CoAP Delete operation on the URI of the topic. The CoAP PubSub Broker MUST return 2.02 Deleted if the REMOVE operation is successful. The broker MUST return the appropriate 4.xx response code indicating the reason for failure if the topic can not be REMOVEd. Figure 5 shows a successful REMOVE of topic1.

The REMOVE interface is specified as follows:

Interaction: Client -> Broker

Method: GET

URI Template: /{+ps}/{topic}

URI Template Variables:

ps := PubSub Function Set path (mandatory). The path of the PubSub Function Set, as obtained from discovery.

topic := The desired topic to REMOVE.

Content-Format: Any valid CoAP Content Format

Response Payload: Representation of the topic value (CoAP resource state representation) in the indicated Content-Format

The following response codes are defined for this interface:

Success: 2.02 "Deleted". Successful REMOVE

Failure: 4.00 "Bad Request". Malformed request.

Failure: 4.01 "Unauthorized". Authorization failure.

Failure: 4.04 "Not Found". Topic does not exist.

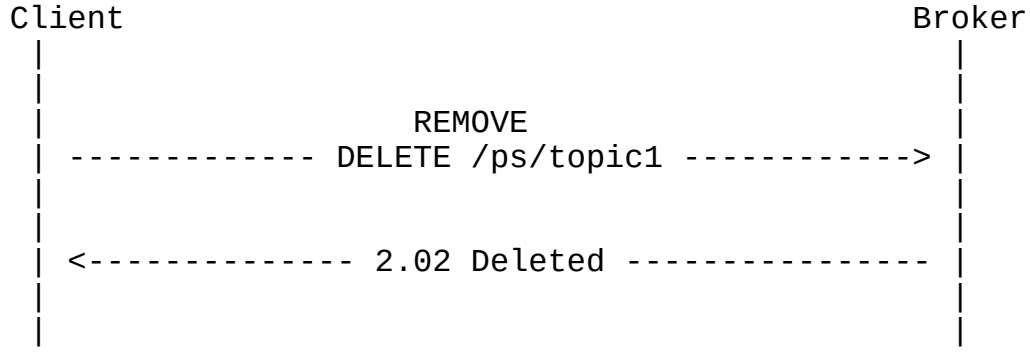

Figure 8: REMOVE

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 which registers 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-Wake Operation

A CoAP PubSub client MAY sleep between PUBLISH operations to a Broker. A CoAP PubSub client MAY sleep between READ operations from a broker. A CoAP PubSub Client SHOULD NOT sleep while SUBSCRIBEd to any topics on a 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 clients SHOULD authenticate each other and enforce access control policies. A malicious client 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 for example a client device on a sensor network and a client application running in a cloud-based server infrastructure since brokers terminate the exchange. While running separate DTLS sessions from the client device to the broker and from broker to client application protects confidentially on those paths, the client device does not know whether the commands coming from the broker are actually coming from the client application. Similarly, a client application requesting data does not know whether the data originated on the client device. 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 device that any request originated at the client application. Similarly, integrity protected sensor data from a client device will also provide guarantee to the client application that the data originated on the client device 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 one attribute value in the Resource Type (rt=) registry established with [RFC6690] and appends to the definition of one CoAP Response Code in the CoRE Parameters Registry.

8.1. Resource Type value 'core.ps'

- Attribute Value: core.ps
- Description: Section 4 of [[This document]]
- Reference: [[This document]]
- Notes: None

8.2. Response Code value '2.04'

- Response Code: 2.04
- Description: Add No Content response to GET to the existing definition of the 2.04 response code.
- Reference: [[This document]]
- Notes: None

9. Acknowledgements

The authors would like to thank Hannes Tschofenig, Zach Shelby, Mohit Sethi, Peter Van der Stok, Tim Kellogg, 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.

[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.

[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