Workgroup: CoRE Working Group
Internet-Draft: draft-ietf-core-coap-pubsub-13
Published: 20 October 2023
Intended Status: Standards Track
Expires: 22 April 2024
Authors: J. Jimenez
Ericsson
M. Koster
Dogtiger Labs
A. Keranen
Ericsson

A publish-subscribe architecture for the Constrained Application Protocol (CoAP)

Abstract

This document describes a publish-subscribe architecture for the Constrained Application Protocol (CoAP), extending the capabilities of CoAP communications for supporting endpoints with long breaks in connectivity and/or up-time. CoAP clients publish on and subscribe to a topic via a corresponding topic resource at a CoAP server acting as broker.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-core-coap-pubsub/.

Discussion of this document takes place on the core Working Group mailing list (mailto:core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/. Subscribe at https://www.ietf.org/mailman/listinfo/core/.

Source for this draft and an issue tracker can be found at https://github.com/core-wg/coap-pubsub.

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 https://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 22 April 2024.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] supports machine-to-machine communication across networks of constrained devices and constrained networks. 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, a client, or both.

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 up-time because they spend most of their time in a sleeping state with no network connectivity. Another important class of nodes are devices with limited reachability due to middle-boxes like Network Address Translators (NATs) and firewalls.

For these nodes, the client/server-oriented architecture of REST can be challenging when interactions are not initiated by the devices themselves. A publish/subscribe-oriented architecture where nodes exchange data via topics through a broker entity might fit these nodes better.

This document applies the idea of broker-based publish-subscribe to Constrained RESTful Environments using CoAP. It defines a broker that allows to create, discover subscribe and publish on topics.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification requires readers to be familiar with all the terms and concepts that are discussed in [RFC8288] and [RFC6690]. Readers should also be familiar with the terms and concepts discussed in [RFC7252], [RFC9176] and [RFC7641]. The URI template format [RFC6570] is used to describe the REST API defined in this specification.

This specification makes use of the following terminology:

publish-subscribe (pub/sub):
: A message communication model where messages associated with specific topics are sent to a broker. Interested parties, i.e. subscribers, receive these topic-based messages from the broker without the original sender knowing the recipients. The broker handles matching and delivering these messages to the appropriate subscribers.

publishers and subscribers:
: CoAP clients can act as publishers or as subscribers. Publishers send CoAP messages (publications) to the broker on specific topics. Subscribers have an ongoing observation relation (subscription) to a topic. Both roles operate without any mutual knowledge, guided by their respective topic interests.

topic collection:
: A set of topic configurations. A topic collection is hosted as one collection resource at the broker, and its representation is the list of links to the topic resources corresponding to each topic configuration.

topic-configuration:
: A set of information concerning a topic, including its configuration and other metadata. A topic configurations is hosted as one topic resource at the broker, and its representation is the set of configuration information concerning the topic. All the topic resources associated with the same topic collection share a common base URI, i.e., the URI of the collection resource. Throughout this document the word "topic" and "topic-configuration" can be used interchangeably.

topic-data resource:
: A resource where clients can publish data and/or subscribe to data for a specific topic. The representation of the topic resource corresponding to such a topic also specifies the URI to the present topic-data resource.

broker:
: A CoAP server that hosts one or more topic collections with their topic-configurations, and possibly also topic-data resources. The

broker is responsible for the store-and-forward of state update representations, for the topics for which it hosts the corresponding topic-data resources. The broker is also responsible of handling the topic lifecycle as defined in Section 3.1. The creation, configuration, and discovery of topics at a broker is specified in Section 2.

1.2. CoAP Publish-Subscribe Architecture

Figure 1 shows a simple Publish/Subscribe architecture over CoAP.

Topics are created by the broker, but the initial configuration can be proposed by a client (e.g., a publisher or a dedicated administrator) over the RESTful interface of a corresponding topic resource hosted by the broker.

Publishers submit their data over the RESTful interface of a topic-data resource corresponding to the topic, which may be hosted by the broker. Subscribers to a topic are notified of new publications by using Observe [RFC7641] on the corresponding topic-data resource.

The broker is responsible for the store-and-forward of state update representations between CoAP clients. Subscribers observing a resource will receive notifications, the delivery of which is done on a best-effort basis.

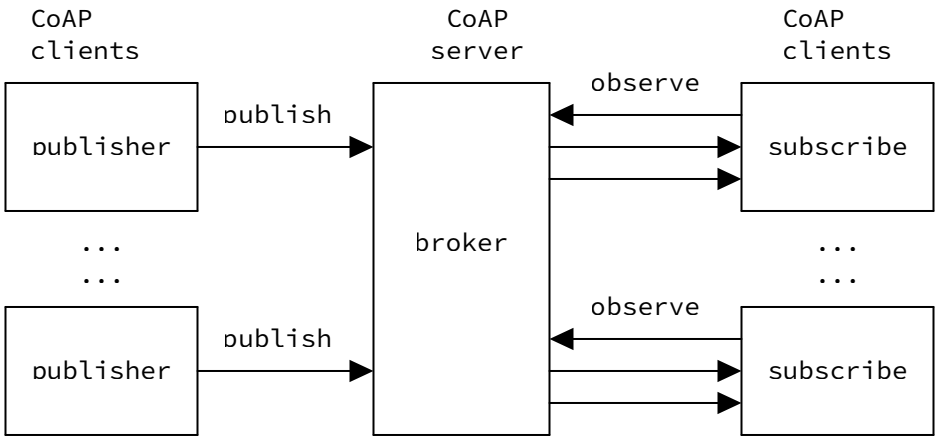

Figure 1: Publish-subscribe architecture over CoAP

This document describes two sets of interactions, interactions to configure topics and their lifecycle (see Section 2.5) and interactions about the topic-data (see Section 3.2).

Topic-configuration interactions are discovery, create, read configuration, update configuration, delete configuration and handle the management of the topics.

Topic-data interactions are publish, subscribe, unsubscribe, read and delete, these operations are oriented on how data is transferred from a publisher to a subscriber.

1.3. Managing Topics

Figure 2 shows the resources related to a Topic Collection that can be managed at the Broker.

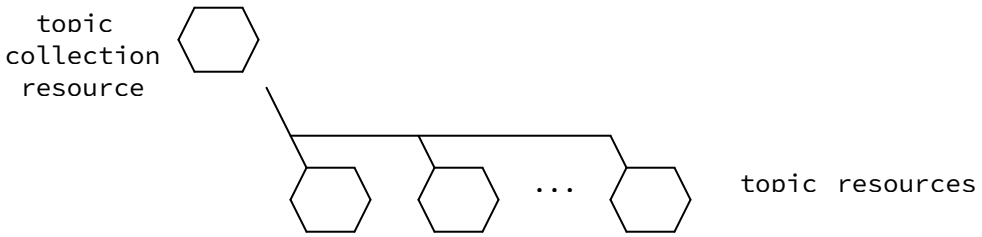

Figure 2: Resources of a Broker

The Broker exports one or more topic-collection resources, with resource type "core.ps.coll" defined in Section 6 of this document. The interfaces for the topic-collection resource is defined in Section 2.4.

A topic-collection resource can have topic resources as its children resources, with resource type "core.ps.conf".

2. Pub-Sub Topics

The configuration side of a "publish/subscribe broker" consists of a collection of topics. These topics as well as the collection itself are exposed by a CoAP server as resources (see Figure 3). Each topic is associated with: a topic resource and a a topic-data resource. The topic resource is used by a client creating or administering a topic. The topic-data resource is used by the publishers and the subscribers to a topic.

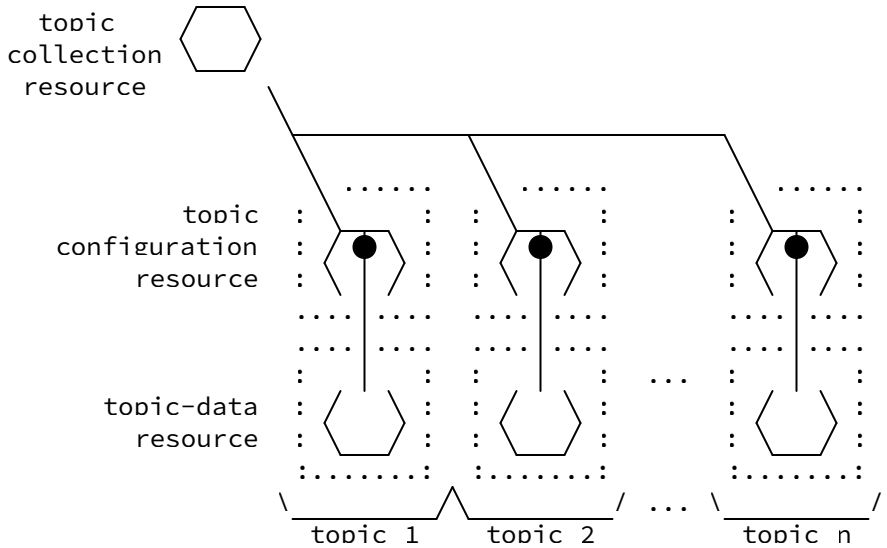

Figure 3: Topic and topic-data resources of a topic

2.1. Collection Representation

Each topic configuration is represented as a link, where the link target is the URI of the corresponding topic resource.

Publication and subscription to a topic occur at a link, where the link target is the URI of the corresponding topic-data resource. Such a link is specified by the topic-data entry within the topic resource (see Section 2.2.1).

A topic resource with a topic-data link can also be simply called "topic".

The list of links to the topic resources can be retrieved from the associated topic collection resource, and represented as a Link Format document [RFC6690]where each such link specifies the link target attribute 'rt' (Resource Type), with value "core.ps.conf" defined in this document.

2.2. Topic-Configuration Representation

A CoAP client can create a new topic by submitting an initial configuration for the topic (see Section 2.4.3). It can also read and update the configuration of existing topics and delete them when they are no longer needed (see Section 2.5).

The configuration of a topic itself consists of a set of properties that can be set by a client or by the broker. The topic-

configuration is represented as a CBOR map containing the configuration properties of the topic as top-level elements.

Unless specified otherwise, these are defined in this document and their CBOR abbreviations are defined in Section 4.

2.2.1. Topic Properties

The CBOR map includes the following configuration parameters, whose CBOR abbreviations are defined in Section 4 of this document.

* 'topic-name': A required field used as an application identifier. It encodes the topic name as a CBOR text string. Examples of topic names include human-readable strings (e.g., "room2"), UUIDs, or other values.

* 'topic-data': A required field (optional during creation) containing the URI of the topic-data resource for publishing/ subscribing to this topic. It encodes the URI as a CBOR text string.

* 'resource-type': A required field used to indicate the resource type of the topic-data resource for the topic. It encodes the resource type as a CBOR text string. The value should be "core.ps.conf".

* 'media-type': An optional field used to indicate the media type of the topic-data resource for the topic. It encodes the media type as a this information as the integer identifier of the CoAP content-format (e.g., value is "50" for "application/json").

* 'topic-type': An optional field used to indicate the attribute or property of the topic-data resource for the topic. It encodes the attribute as a CBOR text string. Example attributes include "temperature".

* 'expiration-date': An optional field used to indicate the expiration date of the topic. It encodes the expiration date as a CBOR text string. The value should be a date string in ISO 8601 format (e.g., "2023-03-31T23:59:59Z"). The broker can use this field to automatically remove topics that are no longer valid. If this field is not present, the topic will not expire automatically.

* 'max-subscribers': An optional field used to indicate the maximum number of simultaneous subscribers allowed for the topic. It encodes the maximum number as an unsigned CBOR integer. If this field is not present, there is no limit to the number of simultaneous subscribers allowed. The broker can use this field to limit the number of subscribers for the topic.

*'observer-check': An optional field that controls the maximum number of seconds between two consecutive Observe notifications sent as Confirmable messages to each topic subscriber. Encoded as a CBOR unsigned integer greater than 0, it ensures subscribers who have lost interest and silently forgotten the observation do not remain indefinitely on the server's observer list. If another CoAP server hosts the topic-data resource, that server is responsible for applying the observer-check value. The default value for this field is 86400, as defined in [RFC7641], which corresponds to 24 hours.

2.3. Discovery

A client can perform a discovery of: the broker; the topic collection resources and topic resources hosted by the broker; and the topic-data resources associated with those topic resources.

2.3.1. Broker Discovery

CoAP clients MAY discover brokers by using CoAP Simple Discovery, via multicast, through a Resource Directory (RD) [RFC9176] or by other means specified in extensions to [RFC7252]. Brokers MAY register with a RD by following the steps on Section 5 of [RFC9176] with the resource type set to "core.ps" as defined in Section 6 of this document.

The following example shows an endpoint discovering a broker using the "core.ps" resource type over a multicast network. Brokers within the multicast scope will answer the query.

```
=> 0.01 GET
   Uri-Path: coap://[ff0x::fe]/.well-known/core
   Resource-Type: core.ps

<= 2.05 Content
   Payload:
   Content-Format: 40 (application/link-format)
   <coap://mythinguri.com/broker/v1>; rt=core.ps
```

2.3.2. Topic Collection Discovery

A Broker SHOULD offer a topic discovery entry point to enable clients to find topics of interest. The resource entry point is the topic collection resource collecting the topic configurations for those topics (see Section 1.2.2 of [RFC6690]) and is identified by the resource type "core.ps.coll".

The specific resource path is left for implementations, examples in this document use the "/ps" path. The interactions with a topic collection are further defined in Section 2.4.

Since the representation of the topic collection resource includes the links to the associated topic resources, it is not required to locate those links under "/.well-known/core", also in order to limit the size of the Link Format document returned as result of the discovery.

Example:

```
=> 0.01 GET
   Uri-Path: .well-known/core
   Resource-Type: core.ps.coll

   <= 2.05 Content
   Content-Format: 40 (application/link-format)
   </ps1>;rt="core.ps.coll";ct=40,
   </other/path>;rt="core.ps.coll";ct=40
```

2.3.3. Topic-Configuration Discovery

Each topic collection is associated with a group of topic resources, each detailing the configuration of its respective topic (refer to Section 2.2.1). Each topic resource is identified by the resource type "core.ps.conf".

Below is an example of discovery via /.well-known/core with rt=core.ps.conf that returns a list of topics, as the list of links to the corresponding topic resources.

```
=> 0.01 GET
   Uri-Path: .well-known/core
   Resource-Type: core.ps.conf

<= 2.05 Content
   Content-Format: 40 (application/link-format)
   </ps1/h9392>;rt="core.ps.conf";ct=TBD,
   </other/path/2e3570>;rt=core.ps.conf;ct=TBD
```

2.3.4. Topic-Data Discovery

Within a topic, there is the topic-data property containing the URI of the topic-data resource that a CoAP client can subscribe and publish to. Resources exposing resources of the topic-data type are expected to use the resource type 'core.ps.data'.

The topic-data contains the URI of the topic-data resource for publishing and subscribing. So retrieving the topic configuration will also provide the URL of the topic-data (see Section 2.5.1).

It is also possible to discover a list of topic-data resources by sending a request to the collection with with rt=core.ps.data resources as shown below.

```
=> 0.01 GET
   Uri-Path: /ps
   Resource-Type: core.ps.data

<= 2.05 Content
   Content-Format: 40 (application/link-format)
   </ps/data/62e4f8d>; rt=core.ps.data; obs
```

2.4. Topic Collection Interactions

These are the interactions that can happen directly with a specific topic collection.

2.4.1. Retrieving all topic-configurations

A client can request a collection of the topics present in the broker by making a GET request to the collection URI.

On success, the server returns a 2.05 (Content) response, specifying the list of links to topic resources associated with this topic collection (see Section 2.2).

Depending on its granted permissions, a client MAY retrieve a different list of links, corresponding to the topics that the client is authorized to access.

Example:

```
=> 0.01 GET
   Uri-Path: ps

<= 2.05 Content
   Content-Format: 40 (application/link-format)
   </ps/h9392>;rt="core.ps.conf",
   </ps/2e3570>; rt="core.ps.conf"
```

2.4.2. Getting topic-configurations by Properties

A client can filter a collection of topics by submitting the representation of a topic filter (see Section 2.5.2) in a FETCH request to the topic collection URI.

On success, the server returns a 2.05 (Content) response with a representation of a list of topics in the collection (see Section 2.3.3) that match the filter in CoRE link format [RFC6690].

Upon success, the server responds with a 2.05 (Content), providing a list of links to topic resources associated with this topic collection that match the request's filter criteria (refer to Section 2.3.3). A positive match happens only when each request parameter is present with the indicated value in the topic resource representation.

Example:

```
=> 0.05 FETCH
   Uri-Path: ps
   Content-Format: TBD (application/pubsub+cbor)

   {
     "resource-type" : "core.ps.conf"
     "topic-type" : "temperature"
   }

<= 2.05 Content
   Content-Format: 40 (application/link-format)
   </ps/2e3570>;rt="core.ps.conf"
```

2.4.3. Creating a Topic

A client can add a new topic-configurations to a collection of topics by submitting an initial representation of the initial topic resource (see Section 2.2) in a POST request to the topic collection URI. The request MUST specify at least a subset of the properties in Section 2.2.1, namely: topic-name and resource-type.

Please note that the topic will NOT be fully created until a publisher has published some data to it (See Section 3.1).

On success, the server returns a 2.01 (Created) response, indicating the Location-Path of the new topic and the current representation of the topic resource. The response payload includes a CBOR map with key-value pairs. The response must include the required topic properties (see Section 2.2.1), namely: "topic-name", "resource-type" and "topic-data". It may also include a number of optional properties too.

If requirements are defined for the client to create the topic as requested and the broker does not successfully assess that those requirements are met, then the broker MUST respond with a 4.03 (Forbidden) error. The response MUST have Content-Format set to "application/core-pubsub+cbor".

The broker MUST issue a 4.00 (Bad Request) error if a received parameter is invalid, unrecognized, or if the topic-name is already in use or otherwise invalid.

```
=> 0.02 POST
   Uri-Path: ps
   Content-Format: TBD2 (application/core-pubsub+cbor)
   TBD (this should be a CBOR map with the mandatory parameters)
   {
     "topic-name" : "living-room-sensor"
     "resource-type" : "core.ps.conf"
   }

<= 2.01 Created
   Location-Path: ps/h9392
   Content-Format: TBD2 (application/core-pubsub+cbor)

   TBD (this should be a CBOR map)
   {
     "topic-name" : "living-room-sensor",
     "topic-data" : "ps/data/1bd0d6d"
     "resource-type" : "core.ps.conf"
   }
```

2.5. Topic-Configuration Interactions

These are the interactions that can happen at the topic resource level.

2.5.1. Getting a topic-configuration

A client can read the configuration of a topic by making a GET request to the topic resource URI.

On success, the server returns a 2.05 (Content) response with a partial representation of the topic resource, as specified in Section 2.2. The partial representation includes only the configuration parameters such that they are present and have the same value in both the current topic configuration as well as in the FETCH request.

If requirements are defined for the client to create the topic as requested and the broker does not successfully assess that those requirements are met, then the broker MUST respond with a 4.03 (Forbidden) error.

The response payload is a CBOR map, whose possible entries are specified in Section 2.2 and use the same abbreviations defined in Section 4.

For example, below is a request on the topic "ps/h9392":

```
=> 0.01 GET
   Uri-Path: ps
   Uri-Path: h9392

<= 2.05 Content
   Content-Format: TBD2 (application/core-pubsub+cbor)
   {
      "topic-name" : "living-room-sensor",
      "topic-data" : "ps/data/1bd0d6d",
      "resource-type": "core.ps.conf",
      "media-type": "application/senml-cbor",
      "topic-type": "temperature",
      "expiration-date": "2023-04-00T23:59:59Z",
      "max-subscribers": 100
   }
```

2.5.2. Getting part of a topic-configuration

A client can read the configuration of a topic by making a FETCH request to the topic resource URI with a filter for specific parameters. This is done in order to retrieve part of the current topic resource.

The request contains a CBOR map with a configuration filter or 'conf-filter', a CBOR array with CBOR abbreviation. Each element of the array specifies one requested configuration parameter of the current topic resource (see Section 2.2).

On success, the server returns a 2.05 (Content) response with a representation of the topic resource. The response has as payload the partial representation of the topic resource as specified in Section 2.2.

If requirements are defined for the client to create the topic as requested and the broker does not successfully assess that those requirements are met, then the broker MUST respond with a 4.03 (Forbidden) error.

The response payload is a CBOR map, whose possible entries are specified in Section 2.2 and use the same abbreviations defined in Section 4.

Both request and response MUST have Content-Format set to "application/core-pubsub+cbor".

Example:

```
=> 0.05 FETCH
   Uri-Path: ps
   Uri-Path: h9392
   Content-Format: TBD2 (application/core-pubsub+cbor)
   {
     "conf-filter" : [topic-data, media-type]
   }

<= 2.05 Content
   Content-Format: TBD2 (application/core-pubsub+cbor)
   {
     "topic-data" : "ps/data/1bd0d6d",
     "media-type": "application/senml-cbor"
   }
```

2.5.3. Updating the topic-configuration

A client can update a topic's configuration by submitting the updated topic representation in a PUT request to the topic URI. However, the parameters "topic-name", "topic-data", and "resource-type" are immutable post-creation, and any request attempting to change them will be deemed invalid by the broker.

On success, the server returns a 2.04 (Changed) response and the current full resource representation. The broker may chose not to overwrite parameters that are not explicitly modified in the request.

Note that updating the "topic-data" path will automatically cancel all existing observations on it and thus will unsubscribe all subscribers. Similarly, decreasing max-subscribers will also cause that some subscribers get unsubscribed. Unsubscribed endpoints SHOULD receive a final 4.04 (Not Found) response as per [RFC7641] Section 3.2.

Example:

```
=> 0.03 PUT
   Uri-Path: ps
   Uri-Path: h9392
   Content-Format: TBD2 (application/core-pubsub+cbor)

   {
      "topic-name" : "living-room-sensor",
      "topic-data" : "ps/data/1bd0d6d",
      "topic-type": "temperature",
      "expiration-date": "2023-04-28T23:59:59Z",
      "max-subscribers": 2
   }

<= 2.04 Changed
   Content-Format: TBD2 (application/core-pubsub+cbor)

   TBD (this should be a CBOR map)
   {
      "topic-name" : "living-room-sensor",
      "topic-data" : "ps/data/1bd0d6d",
      "resource-type": "core.ps.conf",
      "media-type": "application/senml-cbor",
      "topic-type": "temperature",
      "expiration-date": "2023-04-28T23:59:59Z",
      "max-subscribers": 2
   }
```

Note that when a topic configuration changes, it may result in disruptions for the subscribers. Some potential issues that may arise include:

- Limiting the number of subscribers will cause to cancel ongoing subscriptions until max-subscribers has been reached.
- Changing the topic-data value will cancel all ongoing subscriptions.
- Changing of the expiration-date may cause to cancel ongoing subscriptions if the topic expires at an earlier data.

2.5.4. Deleting a topic-configuration

A client can delete a topic by making a CoAP DELETE request on the topic resource URI.

On success, the server returns a 2.02 (Deleted) response.

When a topic-configuration resource is deleted, the broker MUST also delete the topic-data resource, unsubscribe all subscribers by

removing them from the list of observers and returning a final 4.04 (Not Found) response as per section 3.2 of [RFC7641].

Example:

```
=> 0.04 DELETE
   Uri-Path: ps
   Uri-Path: h9392

<= 2.02 Deleted
```

3. Publish and Subscribe

The overview of the publish/subscribe mechanism over CoAP is as follows: a publisher publishes to a topic by submitting the data in a PUT request to a topic-data resource and subscribers subscribe to a topic by submitting a GET request with Observe option set to 0 (register) to a topic-data resource. When resource state changes, subscribers observing the resource [RFC7641] at that time will receive a notification.

A topic-data resource does not exist until some initial data has been published to it. Before initial data publication, a GET request to the topic-data resource URI results in a 4.04 (Not Found) response. If such a "half created" topic is undesired, the creator of the topic can simply immediately publish some initial placeholder data to make the topic "fully created" (see Section 3.1).

URIs for topic resources are broker-generated (see Section 2.4.3). There is no necessary URI pattern dependence between the URI where the topic-data exists and the URI of the topic-configuration resource.

3.1. Topic Lifecycle

When a topic is newly created, it is first placed by the broker into the HALF CREATED state (see Figure 4). In this state, a client can read and update the configuration of the topic and delete the topic. A publisher can publish to the topic-data resource. However, a subscriber cannot yet subscribe to the topic-data resource nor read the latest data.

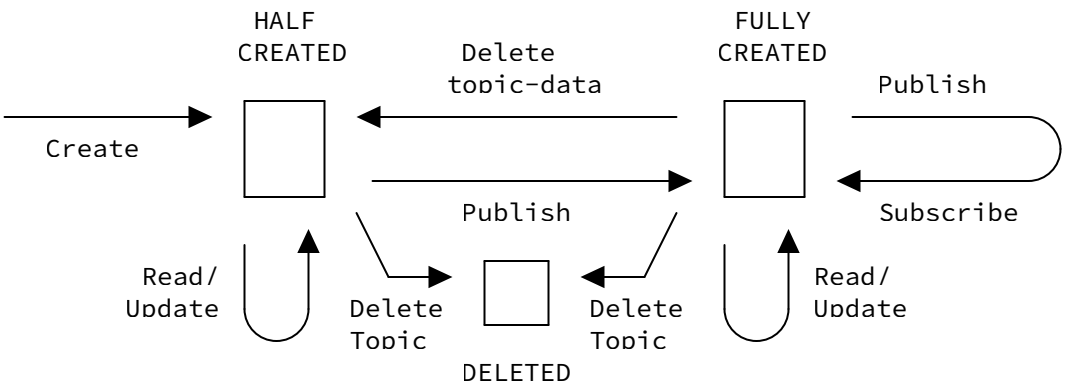

Figure 4: Lifecycle of a Topic

After a publisher publishes to the topic-data for the first time, the topic is placed into the FULLY CREATED state. In this state, a client can read data by means of a GET request without observe. A publisher can publish to the topic-data resource and a subscriber can observe the topic-data resource.

When a client deletes a topic-configuration resource, the topic is placed into the DELETED state and shortly after removed from the server. In this state, all subscribers are removed from the list of observers of the topic-data resource and no further interactions with the topic are possible.

When a client deletes a topic-data, the topic is placed into the HALF CREATED state, where clients can read, update and delete the topic-configuration and await for a publisher to begin publication.

3.2. Topic-Data Interactions

Interactions with the topic-data resource are covered in this section.

3.2.1. Publish

A topic-configuration with a topic-data resource must have been created in order to publish data to it (See Section 2.4.3) and be in the half-created or fully-created state in order to the publish operation to work (see Section 3.1).

A client can publish data to a topic by submitting the data in a PUT request to the topic-data URI as indicated in its topic resource property. Please note that the topic-data URI is not the same as the topic-configuration URI used for configuring the topic (see Section 2.2).

On success, the server returns a 2.04 (Updated) response. However, when data is published to the topic for the first time, the server instead MUST return a 2.01 (Created) response and set the topic in the fully-created state (see Section 3.1).

If the request does not have an acceptable content-format, the server returns a 4.15 (Unsupported Content-Format) response.

If the client is sending publications too fast, the server returns a 4.29 (Too Many Requests) response [RFC8516].

Example of first publication:

```
=> 0.03 PUT
   Uri-Path: ps
   Uri-Path: data
   Uri-Path: 1bd0d6d
   Content-Format: 110

   {
      "n": "temperature",
      "u": "Cel",
      "t": 1621452122,
      "v": 23.5
   }

<= 2.01 Created
```

Example of subsequent publication:

```
=> 0.03 PUT
   Uri-Path: ps
   Uri-Path: data
   Uri-Path: 1bd0d6d
   Content-Format: 110

   {
      "n": "temperature",
      "u": "Cel",
      "t": 182734122,
      "v": 22.5
   }

<= 2.04 Updated
```

3.2.2. Subscribe

A client can subscribe to a topic-data by sending a CoAP GET request with the Observe set to 0 to subscribe to resource updates. [RFC7641].

On success, the server hosting the topic-data resource MUST return 2.05 (Content) notifications with the data and the Observe Option. Otherwise, if no Observe Option is present the client should assume that the subscription was not successful.

If the topic is not yet in the fully created state (see Section 3.1) the broker SHOULD return a response code 4.04 (Not Found).

The following response codes are defined for the Subscribe operation:

Success: 2.05 "Content". Successful subscribe with observe response, current value included in the response.

Failure: 4.04 "Not Found". The topic-data does not exist.

If the 'max-subscribers' parameter has been reached, the server must treat that as specified in section 4.1 of [RFC7641]. The response MUST NOT include an Observe Option, the absence of which signals to the subscriber that the subscription failed.

Example:

```
=> 0.01 GET
   Uri-Path: ps
   Uri-Path: data
   Uri-Path: 1bd0d6d
   Observe: 0

<= 2.05 Content
   Content-Format: 110
   Observe: 10001
   Max-Age: 15

  {
    "bn": "urn:dev:os:193-iot/sparrow/jorvas/",
    "n": "Raitis-lampotila",
    "u": "Cel",
    "t": 1696340182,
    "v": 19.87
  }

<= 2.05 Content
   Content-Format: 110
   Observe: 10002
   Max-Age: 15

  {
    "bn": "urn:dev:os:193-iot/sparrow/jorvas/",
    "n": "Raitis-lampotila",
    "u": "Cel",
    "t": 1696340182,
    "v": 21.87
  }
```

3.2.3. Unsubscribe

A CoAP client can unsubscribe simply by cancelling the observation as described in Section 3.6 of [RFC7641]. The client MUST either use CoAP GET with the Observe Option set to 1 or send a CoAP Reset message in response to a notification. Also on Section 3.6 of [RFC7641] the client can simply "forget" the observation and the server will remove it from the list of observers after the next notification.

As per [RFC7641] a server that transmits notifications mostly in non-confirmable messages, but it MUST send a notification in a confirmable message instead of a non-confirmable message at least every 24 hours.

This value can be modified at the broker by the administrator of a topic by modifying the parameter "observer-check" on Section 2.2. This would allow to change the rate at which different

implementations verify that a subscriber is still interested in observing a topic-data resource.

3.2.4. Delete topic-data

A publisher MAY delete a topic by making a CoAP DELETE request on the topic-data URI.

On success, the server returns a 2.02 (Deleted) response.

When a topic-data resource is deleted, the broker SHOULD also delete the topic-data parameter in the topic resource, unsubscribe all subscribers by removing them from the list of observers and return a final 4.04 (Not Found) response as per [RFC7641] Section 3.2. The topic is then set back to the half created state as per Section 3.1.

Example:

```
=> 0.04 DELETE
   Uri-Path: ps
   Uri-Path: data
   Uri-Path: 1bd0d6d

<= 2.02 Deleted
```

3.3. Read latest data

A client can get the latest published topic-data by making a GET request to the topic-data URI in the broker. Please note that discovery of the topic-data parameter is a required previous step (see Section 2.5.1).

On success, the server MUST return 2.05 (Content) response with the data.

If the target URI does not match an existing resource or the topic is not in the fully created state (see Section 3.1), the broker MUST return a response code 4.04 (Not Found).

Example:

```
=> 0.01 GET
   Uri-Path: ps
   Uri-Path: data
   Uri-Path: 1bd0d6d

<= 2.05 Content
   Content-Format: 110
   Max-Age: 15

   {
      "n": "temperature",
      "u": "Cel",
      "t": 1621452122,
      "v": 23.5
   }
```

3.4. Rate Limiting

The server hosting the topic-data may have to handle a potentially large number of publishers and subscribers at the same time. This means it could become overwhelmed if it receives too many publications in a short period of time.

In this situation, if a publisher is sending publications too fast, the server SHOULD return a 4.29 (Too Many Requests) response [RFC8516]. As described in [RFC8516], the Max-Age option [RFC7252] in this response indicates the number of seconds after which the client may retry. The broker MAY also stop publishing messages from that publisher for the indicated time.

When a publisher receives a 4.29 (Too Many Requests) response, it MUST NOT send any new publication requests to the same topic-data resource before the time indicated by the Max-Age option has passed.

4. CoAP Pubsub Parameters

This document defines parameters used in the messages exchanged between a client and the broker during the topic creation and configuration process (see Section 2.2). The table below summarizes them and specifies the CBOR key to use instead of the full descriptive name.

Note that the media type application/core-pubsub+cbor MUST be used when these parameters are transported in the respective message fields.

Name	CBOR Key	CBOR Type	Reference
topic-name	TBD1	tstr	[RFC-XXXX]
topic-data	TBD2	tstr	[RFC-XXXX]
resource-type	TBD3	tstr	[RFC-XXXX]
media-type	TBD4	uint	[RFC-XXXX]
topic-type	TBD5	tstr	[RFC-XXXX]
expiration-date	TBD6	tstr	[RFC-XXXX]
max-subscribers	TBD7	uint	[RFC-XXXX]
observer-check	TBD8	uint	[RFC-XXXX]

Figure 5: CoAP Pubsub Parameters

5. Security Considerations

The architecture presented in this document inherits the security considerations from CoAP [RFC7252] and Observe [RFC7641], as well as from Web Linking [RFC8288], Link-Format [RFC6690], and the CoRE Resource Directory [RFC9176].

Communications between each client and the broker MUST be secured, e.g., by using OSCORE [RFC8613] or DTLS [RFC9147]. Security considerations for the used secure communication protocols apply too.

The content published on a topic by a publisher client SHOULD be protected end-to-end between the publisher and all the subscribers to that topic. In such a case, it MUST be possible to assert source authentication of the published data. This can be achieved at the application layer, e.g., by using COSE [RFC9052], [RFC9053], [RFC9338].

Access control of clients at the broker MAY be enforced for performing discovery operation, and SHOULD be enforced in a fine-grained fashion for operations related to the the creation, update, and deletion of topic resources, as well as for operations on topic-data resources such as publication on and subscription to topics. This prevents rogue clients to, among other things, repeatedly create topics at the broker or publish (large) contents, which may result in Denial of Service against the broker and the active subscribers.

Building on [I-D.ietf-ace-key-groupcomm], its application profile for publish-subscribe communication with CoAP [I-D.ietf-ace-pubsub-profile] provides a security model that can be used in the architecture presented in this document, in order to enable secure communication between the different parties as well as

secure, authorized operations of publishers and subscribers that fulfill the requirements above.

In particular, the application profile above relies on the ACE framework for Authentication and Authorization in Constrained Environments (ACE) [RFC9200] and defines a method to: authorize publishers and subscribers to perform operations at the broker, with fine-grained access control; authorize publishers and subscribers to obtain the keying material required to take part to a topic managed by the broker; protect published data end-to-end between its publisher and all the subscribers to the targeted topic, ensuring confidentiality, integrity, and source authentication of the published content end-to-end. That approach can be extended to enforce authorization and fine-grained access control for administrator clients that are intended to create, update, and delete topic configurations at the broker.

6. IANA Considerations

This document has the following actions for IANA.

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

6.1. Media Type

IANA is requested to add the following Media-Type to the "Media Types" registry [IANA.media-types].

Name	Template	Reference
pubsub+cbor	application/pubsub+cbor	RFC XXXX, Section 6.1

Table 1: New Media Type application/pubsub+cbor

Type name: application
Subtype name: pubsub+cbor
Required parameters: N/A
Optional parameters: N/A
Encoding considerations: binary (CBOR data item)
Security considerations: Section 5 of RFC XXXX
Interoperability considerations: none
Published specification: Section 6.1 of RFC XXXX
Applications that use this media type: This type is used by clients that create, retrieve, and update topic configurations at servers acting as a pub-sub broker.
Fragment identifier considerations: N/A
Person & email address to contact for further information: CoRE WG mailing list (core@ietf.org), or IETF Applications and Real-Time Area (art@ietf.org)
Intended usage: COMMON

Restrictions on usage: none
Author/Change controller: IETF
Provisional registration: no

6.2. Content-Format

IANA has added the following Content-Formats to the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [IANA.core-parameters], as follows:

Content Type	Content Coding	ID	Reference
application/pubsub+cbor	-	TBD9	RFC XXXX

Table 2: New Content-Format

TBD9 is to be assigned from the space 256..999.

6.3. CoAP Pubsub Parameters

IANA is asked to register the following entries in the subregistry of the "Constrained RESTful Environments (CoRE) Parameters" registry group.

This specification establishes the "Pubsub Topic Configuration Parameters" IANA registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

The columns of this registry are:

* Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

* CBOR Key: This is the value used as CBOR key of the item. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 as well as text strings of length 1 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535, as well as text strings of length 2 are designated as "Specification Required". Integer values greater than 65535 as well as text strings of length greater than 2 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* CBOR Type: This contains the CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.

*Reference: This contains a pointer to the public specification for the item.

The registry is initially populated with the entries in Figure 5 of Section 4.

6.4. Resource Types

IANA is asked to enter the following values in the "Resource Type (rt=) Link Target Attribute Values" registry within the "Constrained Restful Environments (CoRE) Parameters" registry group.

Value: core.ps
Description: Publish-Subscribe Broker
Reference: [RFC-XXXX]

Value: core.ps.coll
Description: Topic-collection resource of a Publish-Subscribe Broker
Reference: [RFC-XXXX]

Value: core.ps.conf
Description: Topic-configuration resource of a Publish-Subscribe Broker
Reference: [RFC-XXXX]

Value: core.ps.data
Description: Topic-data resource of a broker
Reference: [RFC-XXXX]

Acknowledgements

The current version of this document contains a substantial contribution by Klaus Hartke's proposal [I-D.hartke-t2trg-coral-pubsub], which defines the topic resource model and structure as well as the topic lifecycle and interactions. It also follows a similar architectural design as that provided by Marco Tiloca's [I-D.ietf-ace-oscore-gm-admin].

The authors would like to also thank Carsten Bormann, Hannes Tschofenig, Zach Shelby, Mohit Sethi, Peter van der Stok, Tim Kellogg, Anders Eriksson, Goran Selander, Mikko Majanen, Olaf Bergmann and Oscar Novo for their valuable contributions and reviews.

References

Normative References

[IANA.core-parameters] IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

[IANA.media-types]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/rfc/rfc6570>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/rfc/rfc6690>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/rfc/rfc7641>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8516] Keranen, A., ""Too Many Requests" Response Code for the Constrained Application Protocol", RFC 8516, DOI 10.17487/RFC8516, January 2019, <https://www.rfc-editor.org/rfc/rfc8516>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/rfc/rfc8613>.

[RFC9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/rfc/rfc9176>.

Informative References

[I-D.hartke-t2trg-coral-pubsub]
Hartke, K., "Publish/Subscribe over the Constrained Application Protocol (CoAP) using the Constrained RESTful Application Language (CoRAL)", Work in Progress, Internet-Draft, draft-hartke-t2trg-coral-pubsub-01, 9 May 2020, <https://datatracker.ietf.org/doc/html/draft-hartke-t2trg-coral-pubsub-01>.

[I-D.ietf-ace-key-groupcomm] Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-17, 6 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-17>.

[I-D.ietf-ace-oscore-gm-admin] Tiloca, M., Höglund, R., Van der Stok, P., and F. Palombini, "Admin Interface for the OSCORE Group Manager", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-gm-admin-09, 1 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-oscore-gm-admin-09>.

[I-D.ietf-ace-pubsub-profile] Palombini, F., Sengul, C., and M. Tiloca, "Publish-Subscribe Profile for Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-pubsub-profile-07, 13 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-pubsub-profile-07>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/rfc/rfc8288>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version

1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[RFC9200] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth)", RFC 9200, DOI 10.17487/RFC9200, August 2022, <https://www.rfc-editor.org/rfc/rfc9200>.

[RFC9338] Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, DOI 10.17487/RFC9338, December 2022, <https://www.rfc-editor.org/rfc/rfc9338>.

Contributors

Marco Tiloca
RISE AB

Email: marco.tiloca@ri.se

Marco offered comprehensive reviews and insightful guidance on the recent iterations of this document. His contributions were particularly notable in the Security Considerations section, among others.

Authors' Addresses

Jaime Jimenez
Ericsson

Email: jaime@iki.fi

Michael Koster
Dogtiger Labs

Email: michaeljohnkoster@gmail.com

Ari Keranen
Ericsson

Email: ari.keranen@ericsson.com