CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

J. Jimenez
Ericsson
M. Koster
Dogtiger Labs
A. Keranen
Ericsson

2 March 2026

A publish-subscribe architecture for the Constrained Application Protocol (CoAP)
draft-ietf-core-coap-pubsub-19

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

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 [BCP14] (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 (pubsub):
: 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 the dispatching of 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 relation (subscription) to a topic via CoAP Observe [RFC7641]. Both roles operate without any mutual knowledge, guided by their respective topic interests.

topic collection:
: A topic collection is hosted as one collection resource at the broker. A collection resource is a resource that contains links to other resources that a client can add or remove; that concept is described more generally in Section 3.1 of [I-D.ietf-core-interfaces].

topic:
A set of information about an entity at the broker, including its configuration and other metadata. A topic is hosted as one topic resource at the broker, whose representation is the set of topic properties concerning the topic. All the topic resources associated with the same topic collection share a common base URI, i.e., the URI of the topic collection resource.

topic property:
A single element of configuration information that is associated with a topic.

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 component that hosts one or more topic collections with their topics, and typically 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 for 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 based on CoAP.

The broker can create its hosted topics and set their initial configurations. Alternatively, topics can be created together with their initial configuration by a client (e.g., a publisher or a dedicated administrator), over the RESTful interface of the topic collection resource hosted by the broker.

The broker is responsible for the store-and-forward of state update representations between CoAP clients. Publishers submit their data over the RESTful interface of a topic-data resource corresponding to the topic, which can be hosted at the broker. Subscribers to a topic are notified of new publications by using Observe [RFC7641] on the corresponding topic-data resource.

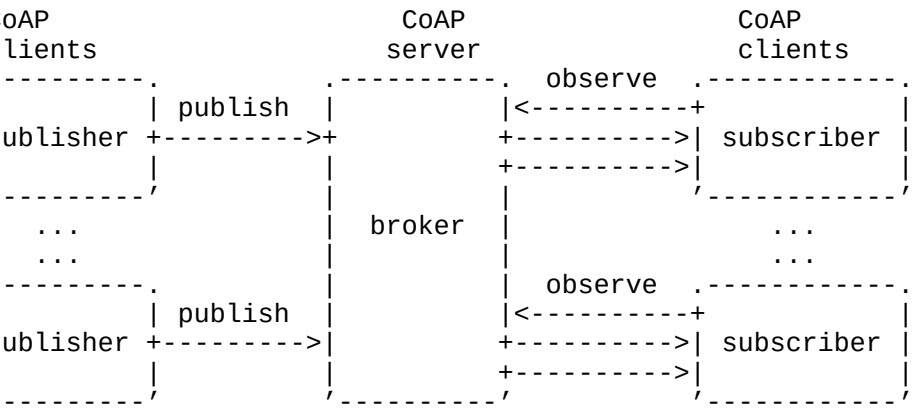

Figure 1: Publish-subscribe architecture based on CoAP

Note that CoAP clients that merely interact with topic configuration but not with topic data (e.g., a dedicated administrator) are not depicted in Figure 1.

This document describes two sets of interactions; interactions to configure topics and their lifecycle (see Section 2.4.3 and Section 2.5) and interactions about the topic-data (see Section 3.2).

Topic interactions are: discovery, create, read configuration, update configuration, and delete configuration. These operations concern the management of the topics.

The 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.

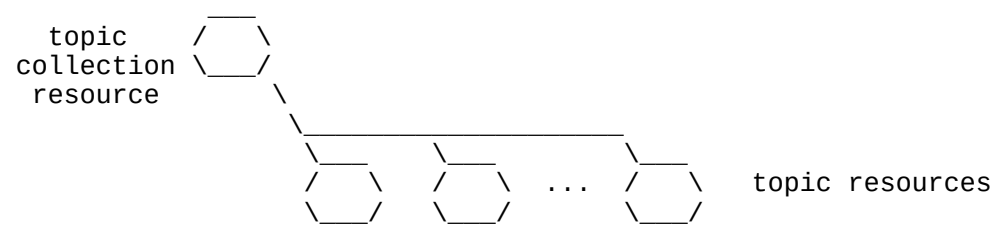

Figure 2: Resources of a Broker

The broker exports one or more topic collection resources, with resource type "core.ps.coll" defined in Section 6.3 of this document. The interface for the topic collection resource is defined in Section 2.4.

A topic collection resource can have topic resources as its child resources, with resource type "core.ps.conf". Other child resource types are currently not defined for a topic collection resource.

2. PubSub Topics

The broker hosts 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 contains a set of properties for configuration, one of which is the URI of the topic-data resource. The topic resource is used by a client for creating or administering a topic. The topic-data resource is used by the publishers and the subscribers to share the data values.

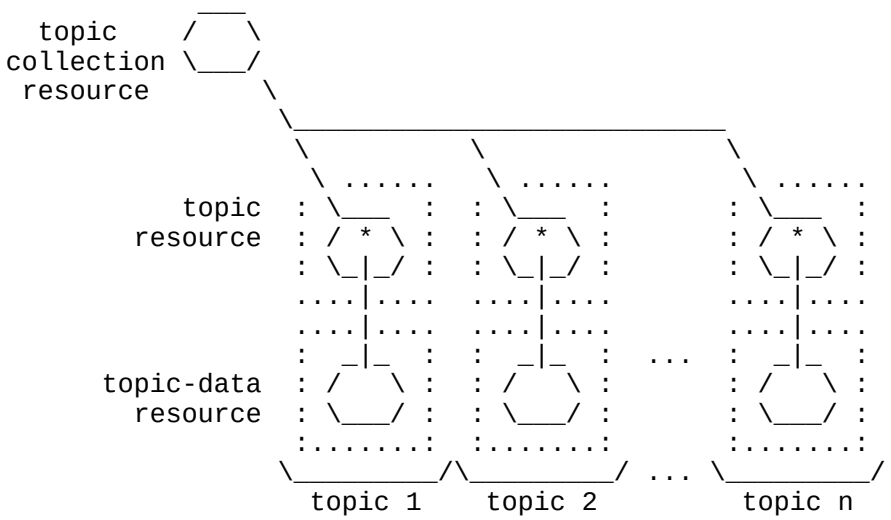

Figure 3: Topic and topic-data resources of a topic

2.1. Collection Representation

Each topic 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 the target of a link, which is the URI of the corresponding topic-data resource. Such a link is specified by the "topic-data" topic property within the topic resource (see Section 2.2.1).

A topic resource can also be simply called "topic".

The list of links to the topic resources can be retrieved from the associated topic collection resource, represented as a CoRE Link Format document [RFC6690] where each link targets a topic resource of type "core.ps.conf" as defined in this document.

2.2. Topic 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 topic properties as well as delete them when they are no longer needed (see Section 2.5).

The configuration of a topic itself consists of a set of topic properties that can be set by a client or by the broker. The topic is represented as a CBOR map containing the topic properties as top-level elements.

Unless specified otherwise, all topic properties 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 topic properties, whose CBOR abbreviations are defined in Section 4.

* "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. The "topic-name" is required at topic creation to enable the broker to detect duplicate creation requests and to provide a stable application-level identifier. Applications that do not require human-readable names MAY use automatically generated values such as UUIDs.

* "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 reference as a CBOR text string. The URI can be that of a resource on a different address than that of the broker; implementations MUST NOT assume that the topic-data resource is co-located with the broker. If a URI is not provided when creating the topic, the choice of the URI for the topic-data resource is left to the broker.

* "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 is typically "core.ps.data", i.e., when using the topic-data resource defined in this document.

* "topic-content-format": This optional field specifies the canonical CoAP Content-Format identifier of the topic-data resource representation as an unsigned integer, e.g., 60 for the media-type "application/cbor".

* "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 tag 1 (epoch-based date/time) as defined in Section 3.4.2 of RFC 8949 [STD94], representing the number of seconds since 1970-01-01T00:00Z in UTC time. If this field is not present, the topic will not expire automatically. When "expiration-date" is reached, the topic resource is deleted as described in Section 2.5.5.

* "max-subscribers": An optional field used to indicate the maximum number of simultaneous subscribers allowed for the topic. It encodes the maximum number as a CBOR unsigned integer. If this field is not present, then there is no limit to the number of simultaneous subscribers allowed. The broker MAY choose to ignore this value if enforcing it would be counterproductive (e.g., causing clients to fall back to polling). This field is intended as a hint from the topic creator; the broker is the final arbiter of resource allocation.

* "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 (see Section 3.2.3). Encoded as a CBOR unsigned integer greater than

0, it ensures that subscribers that 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.

* "initialize": An optional field encoded as a CBOR byte string that contains the initial representation to pre-populate the topic-data resource. When present, the broker MUST create the topic and initialize the topic-data resource with this representation using the Content-Format specified in "topic-content-format". This allows the topic to be immediately subscribable without encountering a 4.04 Not Found error. The representation MUST be valid for the specified Content-Format. For example, for CBOR-based formats, an empty array encoded as 0x80 (CBOR for []) is a valid empty representation. If this field is not present, the broker behaves as usual, and the topic-data resource is not initialized. When this field is present, "topic-content-format" MUST also be specified.

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. Any server implementing a pubsub broker MUST support CoAP discovery with a query parameter as defined in Section 4.1 of [RFC6690] and as used in the examples in this section.

The CoRE Link Format discovery responses shown in the examples in this section are illustrative only. The normative requirements for this format are defined in [RFC6690]. The examples make minimal use of CoRE Link Format attributes, in order to reduce the size of discovery responses: this is beneficial for clients connected to constrained networks. In general, a broker MAY include any CoRE Link Format attributes in each returned link, for example to meet specific use case requirements.

2.3.1. Broker Discovery

CoAP clients MAY discover brokers by using CoAP discovery [RFC7252], via multicast, through a Resource Directory (RD) [RFC9176] or by other means specified in extensions to [RFC7252]. Brokers MAY register with an 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.

Request:

```
Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: ".well-known"
Uri-Path: "core"
Uri-Query: "rt=core.ps"
```

Response:

```
Header: Content (Code=2.05)
Content-Format: 40 (application/link-format)
Payload:
<coaps://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 (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 CoRE Link Format document returned as result of the discovery.

Example:

```
Request:

Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "core"
Uri-Query: "rt=core.ps.coll"

Response:

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

Note that in this example the "ct" attribute is not included for the two collections in the returned CoRE Link Format document. This is because the "ct" attribute is an optional hint, which is not needed in this case: the Content-format of each topic collection resource is implied by its resource type (rt="core.ps.coll") to be 40 ("application/link-format").

2.3.3. Topic Discovery

A broker MAY offer topic resources via /.well-known/core. 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 query rt=core.ps.conf that returns a list of topics, as the list of links to the corresponding topic resources.

```
Request:

Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "core"
Uri-Query: "rt=core.ps.conf"

Response:

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

In certain scenarios, the method described herein may not be applicable, particularly when the server restricts topic availability to authenticated clients only. In such cases, it is recommended to utilize the procedure outlined in Section 2.4.1 and Section 2.4.2 for topic discovery.

2.3.4. Topic-Data Discovery

Within a topic, there is the "topic-data" topic property that contains the URI of the topic-data resource used for publishing and subscribing. So retrieving the topic will also provide the URL of the topic-data resource (see Section 2.5.1).

The topic-data resources use the resource type ’core.ps.data’. It is also possible to discover a list of topic-data resources, by sending a request to the collection resource with a query parameter rt=core.ps.data as shown below. Every topic collection resource MUST support this query.

Request:

```
Header: GET (Code=0.01)
Uri-Path: "ps"
Uri-Query: "rt=core.ps.data"
```

Response:

```
Header: Content (Code=2.05)
Content-Format: 40 (application/link-format)
Payload:
</ps/data/62e4f8d>
```

Note that the "rt" attribute is not included in the returned link in the example response. This is because the query in the request already constrains all links in the response to be only of type "core.ps.data". Therefore, including the "rt" attribute for each returned link would be unnecessary and would make the response size much larger. So the broker, in this example case, was implemented to elide these attributes always, to minimize the size of discovery response payloads.

2.4. Topic Collection Interactions

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

The CoRE Link Format responses shown in the examples in this section are illustrative only. The normative requirements for this format are defined in [RFC6690]. The examples make minimal use of CoRE Link Format attributes, in order to reduce the size of responses: this is beneficial for clients connected to constrained networks. In general, a broker MAY include any CoRE Link Format attributes in each returned link, for example to meet specific use case requirements.

2.4.1. Retrieving all topics

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

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

A client MAY retrieve a list of links to topics it is authorized to access, based on its permissions. A broker MUST implement topic collection discovery.

The content-format 40 ("application/link-format") MUST be supported for the topic collection resource.

Example:

Request:

```
Header: GET (Code=0.01)
Uri-Path: "ps"
```

Response:

```
Header: Content (Code=2.05)
Content-Format: 40 (application/link-format)
Payload:
</ps/h9392>,</ps/2e3570>
```

Note that no "rt" or "ct" attributes are returned for the topic resources in the example payload, because the resource type (rt="core.ps.conf") is already implied by this specification for the topic collection and the content-format (TBD606) is implied by the resource type.

2.4.2. Getting Topics by Topic 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 broker 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 broker 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 topic property in the request payload is present with the indicated value in the topic resource representation.

Example:

Request:

```
Header: FETCH (Code=0.05)
Uri-Path: "ps"
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload:
{
   / topic-name /         0: "temperature",
   / resource-type /      2: "core.ps.data"
}
```

Response:

```
Header: Content (Code=2.05)
Content-Format: 40 (application/link-format)
Payload:
</ps/2e3570>
```

Note that in this example no "rt" or "ct" attributes are returned in the response payload, since the type of each link (topic resource with rt="core.ps.conf") is implied as the default resource type of each topic resource by this specification. Eliding these attributes helps to minimize the size of the response payload.

2.4.3. Creating a Topic

A client can add a new topic to a collection of topics by submitting an initial representation of the 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 topic 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).

To facilitate immediate subscription and allow subscribers to subscribe to the topic before data has been published, the client can include the "initialize" property containing an initial representation (encoded as a CBOR byte string) along with the "topic-content-format" property. When included, the broker MUST create the topic and pre-populate the topic-data resource with the provided representation, using the specified Content-Format. This ensures RFC7641 compliance by maintaining Content-Format consistency across all notifications. For example, for CBOR SenML (Content-Format 60), the initialize value could be h’80’ (the CBOR encoding of an empty array []).

When "initialize" is omitted, the topic will only be fully created after data is published to it.

On success, the broker 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. 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 topic properties too. The response MUST support Content-Format TBD606 ("application/core-pubsub+cbor"), which is the default.

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 reply with a 4.xx client error response (such as 4.03 Forbidden).

The broker MUST reply with a 4.xx client error response (such as 4.00 Bad Request) if a received parameter is invalid, unrecognized, or if the "topic-name" is already in use or otherwise invalid.

```
Request:

Header: POST (Code=0.02)
Uri-Path: "ps"
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-name /         0: "living-room-sensor",
   / resource-type /      2: "core.ps.data"
}

Response:

Header: Created (Code=2.01)
Location-Path: "ps"
Location-Path: "h9392"
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-name /         0: "living-room-sensor",
   / topic-data /         1: "/ps/data/1bd0d6d",
   / resource-type /      2: "core.ps.data"
}
```

2.5. Topic Interactions

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

2.5.1. Getting a topic

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

On success, the broker returns a 2.05 (Content) response with a representation of the topic resource, as specified in Section 2.2.

If requirements are defined for the client to read the topic as requested and the broker does not successfully assess that those requirements are met, then the broker MUST reply with a 4.xx client error response (such as 4.03 Forbidden).

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":

```
Request:

Header: GET (Code=0.01)
Uri-Path: "ps"
Uri-Path: "h9392"

Response:

Header: Content (Code=2.05)
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-name /            0: "living-room-sensor",
   / topic-data /            1: "/ps/data/1bd0d6d",
   / resource-type /         2: "core.ps.data",
   / topic-content-format /  3: 112,
   / topic-type /            4: "temperature",
   / expiration-date /       5: 1(1680393599),
   / max-subscribers /       6: 100
}
```

2.5.2. Getting part of a topic

A client can read the configuration of a topic by making a FETCH request to the topic resource URI with a filter for specific topic properties. 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 of topic properties, using the same abbreviations defined in Section 4. Each element of the array specifies one requested topic property of the current topic resource (see Section 2.2).

On success, the broker 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 read the topic as requested and the broker does not successfully assess that those requirements are met, then the broker MUST reply with a 4.xx client error response (such as 4.03 Forbidden).

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.

Content-Format TBD606 ("application/core-pubsub+cbor") is mandatory to support for both request and response.

The CBOR map in the response payload includes entries for each topic property specified in the request and available in the topic resource representation.

Example:

Request:

```
Header: FETCH (Code=0.05)
Uri-Path: "ps"
Uri-Path: "h9392"
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / conf-filter / 9: [1, 3]
}
```

Response:

```
Header: Content (Code=2.05)
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-data /            1: "/ps/data/1bd0d6d",
   / topic-content-format /  3: 112
}
```

2.5.3. Updating the topic

A client can update a topic’s configuration by submitting the updated topic representation in a POST request to the topic URI. However, the topic properties "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. Since POST replaces the full resource representation, these immutable properties may be included in the request with their current values.

On success, the topic is overwritten and the broker returns a 2.04 (Changed) response and the current full resource representation. The broker MAY choose not to overwrite topic properties that are not explicitly modified in the request.

Similarly, decreasing "max-subscribers" will also cause that some subscribers get unsubscribed. Unsubscribed endpoints receive a final 4.04 (Not Found) response as per Section 3.2 of [RFC7641]. The specific queue management for unsubscribing is left for implementors.

Please note that when using POST the topic is being overwritten, thus some of the optional topic properties (e.g., "max-subscribers", "observer-check") not included in the POST message will be reset to their default values.

Example:

Request:

```
Header: POST (Code=0.03)
Uri-Path: "ps"
Uri-Path: "h9392"
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-name /            0: "living-room-sensor",
   / topic-data /            1: "/ps/data/1bd0d6d",
   / resource-type /         2: "core.ps.data",
   / topic-content-format /  3: 112,
   / topic-type /            4: "temperature",
   / expiration-date /       5: 1(1682719199)
}
```

Response:

```
Header: Changed (Code=2.04)
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-name /            0: "living-room-sensor",
   / topic-data /            1: "/ps/data/1bd0d6d",
   / resource-type /         2: "core.ps.data",
   / topic-content-format /  3: 112,
   / topic-type /            4: "temperature",
   / expiration-date /       5: 1(1682719199),
   / max-subscribers /       6: 100,
   / observer-check /        7: 86400
}
```

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

* Limiting the number of subscribers will cause cancellation of ongoing subscriptions until "max-subscribers" has been reached.

* Changing of the "expiration-date" may cause cancellation of ongoing subscriptions if the topic expires at an earlier data.

2.5.4. Updating the topic with iPATCH

A client can partially update a topic’s configuration by submitting a partial topic representation in an iPATCH request to the topic URI. The iPATCH request allows for updating only specific fields of the topic while leaving the others unchanged. As with the POST method, the topic properties "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 broker returns a 2.04 (Changed) response and the current full resource representation. The broker only updates topic properties that are explicitly mentioned in the request.

Decreasing "max-subscribers" will also cause some subscribers to get unsubscribed. Unsubscribed endpoints receive a final 4.04 (Not Found) response as per Section 3.2 of [RFC7641].

Contrary to POST, iPATCH operations will explicitly update some topic properties, leaving others unmodified.

```
Request:

Header: iPATCH (Code=0.07)
Uri-Path: "ps"
Uri-Path: "h9392"
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / expiration-date /  5: 1(1709164799),
   / max-subscribers /  6: 5
}

Response:

Header: Changed (Code=2.04)
Content-Format: TBD606 (application/core-pubsub+cbor)
Payload (in CBOR diagnostic notation):
{
   / topic-name /            0: "living-room-sensor",
   / topic-data /            1: "/ps/data/1bd0d6d",
   / resource-type /         2: "core.ps.data",
   / topic-content-format /  3: 112,
   / topic-type /            4: "temperature",
   / expiration-date /       5: 1(1709164799),
   / max-subscribers /       6: 5,
   / observer-check /        7: 86400
}
```

Note that when a topic changes through an iPATCH request, it may result in disruptions for the subscribers. For example, limiting the number of subscribers will cause cancellation of ongoing subscriptions until "max-subscribers" has been reached.

2.5.5. Deleting a topic

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

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

When a topic resource is deleted, the broker MUST also delete the topic-data resource. As a result, the broker unsubscribes 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:

Request:

Header: DELETE (Code=0.04)
Uri-Path: "ps"
Uri-Path: "h9392"

Response:

Header: Deleted (Code=2.02)

3. Publish and Subscribe

The overview of the publish-subscribe mechanism based on 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 the 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 resource exists and the URI of the topic 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.

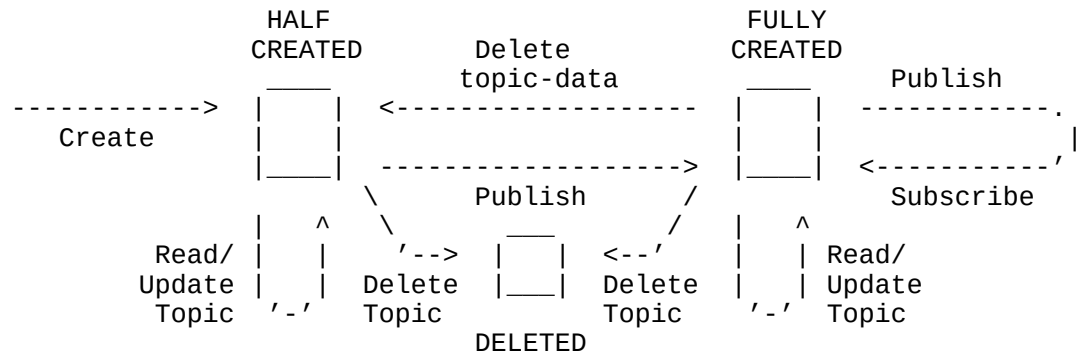

Figure 4: Lifecycle of a Topic

After a publisher publishes to the topic-data resource 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 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. Both the topic resource and the topic-data resource are deleted.

When a client deletes a topic-data resource, the associated topic is placed into the HALF CREATED state, where clients can read, update and delete the topic and await for a publisher to begin publication. the "topic-data" property in the topic configuration remains unchanged but no subscription to topic-data nor reading of data is allowed.

3.2. Topic-Data Interactions

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

3.2.1. Publish

A topic 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 for 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 resource. The URI for this resource is indicated in the "topic-data" topic property value. Please note that this URI is not the same as the topic URI used for configuring the topic (see Section 2.2).

On success, the broker returns a successful response. Typically, this is a 2.04 (Changed) response. However, when data is published to the topic for the first time, the broker returns a 2.01 (Created) response and sets the topic in the fully-created state (see Section 3.1).

Using the "initialize" property is equivalent to having had a first publication with the initial content specified in that property. A follow-up publication from a publisher should result in a 2.04 response from the broker.

If the request does not have an acceptable Content-format, e.g., as specified by the "topic-content-format" property in the topic configuration, the broker returns a 4.15 (Unsupported Content-Format) response.

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

Example of first publication:

Request:

```
Header: PUT (Code=0.03)
Uri-Path: "ps"
Uri-Path: "data"
Uri-Path: "1bd0d6d"
Content-Format: 110
Payload:
{
   "n": "coaps://dev1.example.com/temperature",
   "u": "Cel",
   "t": 1621452122,
   "v": 23.5
}
```

Response:

```
Header: Created (Code=2.01)
```

Example of subsequent publication:

Request:

```
Header: PUT (Code=0.03)
Uri-Path: "ps"
Uri-Path: "data"
Uri-Path: "1bd0d6d"
Content-Format: 110
Payload:
{
   "n": "coaps://dev1.example.com/temperature",
   "u": "Cel",
   "t": 1621452149,
   "v": 22.5
}
```

Response:

```
Header: Changed (Code=2.04)
```

3.2.2. Subscribe

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

On success, the server hosting the topic-data resource returns a successful response (typically 2.05 Content) with the data and the Observe Option. If no Observe Option is present in the response, 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 returns an error response (typically 4.04 Not Found).

The following response codes are defined for the Subscribe operation:

Success: 2.05 "Content". Successful subscription with Observe response, current value included in the response.

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

If the "max-subscribers" topic property value has been reached, the broker must treat that as specified in Section 4.1 of [RFC7641]. The 2.05 (Content) response MUST NOT include an Observe Option, the absence of which signals to the subscriber that the subscription failed.

Example of a successful subscription followed by one update:

```
Request:

Header: GET (Code=0.01)
Uri-Path: "ps"
Uri-Path: "data"
Uri-Path: "1bd0d6d"
Observe: 0

Response:

Header: Content (Code=2.05)
Content-Format: 110
Observe: 10001
Max-Age: 15
Payload:
{
   "n": "urn:dev:os:32473-123456",
   "u": "Cel",
   "t": 1696341182,
   "v": 19.87
}

Response:

Header: Content (Code=2.05)
Content-Format: 110
Observe: 10002
Max-Age: 15
Payload:
{
   "n": "urn:dev:os:32473-123456",
   "u": "Cel",
   "t": 1696340184,
   "v": 21.87
}
```

3.2.3. Unsubscribe

A CoAP client can unsubscribe by canceling the observation as described in Section 3.6 of [RFC7641]. For example, the client can use CoAP GET with the Observe Option set to 1, or simply "forget" the observation and let the server remove it through its own observation lifetime mechanisms.

As per [RFC7641], a server transmits notifications mostly as non-confirmable messages, but it sends a notification as 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 topic property "observer-check" (see Section 2.2). This would allow changing 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 can delete a topic-data resource by making a CoAP DELETE request on the topic-data resource (which is hosted at the "topic-data" URI).

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

When a topic-data resource is deleted, the topic is then set back to the half created state as per Section 3.1 awaiting for a publisher to publish and set the topic to FULLY-CREATED state where clients can subscribe and read the topic-data. The "topic-data" property in the topic configuration remains unchanged, but no subscription to topic-data nor reading of data is allowed.

Note that this is the case irrespective of the value of the "initialize" topic property (if present) in the topic configuration.

Example of a successful deletion:

Request:

Header: DELETE (Code=0.04)
Uri-Path: "ps"
Uri-Path: "data"
Uri-Path: "1bd0d6d"

Response:

Header: Deleted (Code=2.02)

3.3. Read the latest data

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

On success, the server returns a successful response (typically 2.05 Content) 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 returns an error response (typically 4.04 Not Found).

Example:

Request:

```
Header: GET (Code=0.01)
Uri-Path: "ps"
Uri-Path: "data"
Uri-Path: "1bd0d6d"
```

Response:

```
Header: Content (Code=2.05)
Content-Format: 110
Max-Age: 15
Payload:
{
   "n": "coaps://dev1.example.com/temperature",
   "u": "Cel",
   "t": 1621452122,
   "v": 23.5
}
```

As defined in this document, subscribers can retrieve the latest topic-data resource representation. Future specifications can enable subscribers to additionally retrieve old representations of the topic-data resource. The storing of those old representations can be affected, for example, by an additional topic property at the broker that specifies the maximum number of stored old representations of the topic-data. Further details are out of the scope of this document.

3.4. Rate Limiting

The server hosting the topic-data resource 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 dispatching 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. Encoding of PubSub Topic Properties

This document defines topic properties used in the messages exchanged between a client and the broker, for example during the topic creation and configuration process (see Section 2.2). Table 1 summarizes them and specifies the CBOR key that MUST be used instead of the full descriptive name.

Note that the media type application/core-pubsub+cbor MUST be used when these topic properties are transported in the respective CoAP message payloads.

Name	CBOR Key	CBOR Type
topic-name	0	tstr
topic-data	1	tstr
resource-type	2	tstr
topic-content-format	3	uint
topic-type	4	tstr
expiration-date	5	tag 1
max-subscribers	6	uint
observer-check	7	uint
initialize	8	bstr
conf-filter	9	array

Table 1: CoAP Pubsub Topic Properties and CBOR Encoding

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], CoRE Link Format [RFC6690], and the CoRE Resource Directory [RFC9176].

Communications between each client and the broker are RECOMMENDED to 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 [STD96] [RFC9053].

Access control of clients at the broker MAY be enforced for performing discovery operations, and SHOULD be enforced in a fine-grained fashion for operations related to 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 [RFC9594], 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 a 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 topics 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 Registrations

This specification registers the 'application/core-pubsub+cbor' media type for messages of the protocols defined in this document and carrying topic properties encoded in CBOR. This registration follows the procedures specified in [BCP13].

Type name: application

Subtype name: core-pubsub+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Must be encoded as a CBOR map containing the topic properties defined in [RFC-XXXX].

Security considerations: See Section 5 of [RFC-XXXX].

Interoperability considerations: none

Published specification: [RFC-XXXX]

Applications that use this media type: This type is used by clients that create, retrieve, and update topics at servers acting as a broker.

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information: CoRE WG mailing list (core@ietf.org), or IETF Web and Internet Transport (WIT) Area (wit@ietf.org)

Intended usage: COMMON

Restrictions on usage: none

Author/Change controller: IETF

Provisional registration: no

6.2. CoAP Content-Formats

IANA is asked to register the following entry to the "CoAP Content-Formats" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

Content Type: application/core-pubsub+cbor

Content Coding: -

ID: TBD606

Reference: [RFC-XXXX]

6.3. Resource Types

IANA is asked to enter the following values from Table 2 in the "Resource Type (rt=) Link Target Attribute Values" registry within the "Constrained Restful Environments (CoRE) Parameters" registry group. Reference should always be [RFC-XXXX].

Value	Description
core.ps	Publish-subscribe broker
core.ps.coll	Topic collection resource of a publish-subscribe broker
core.ps.conf	Topic resource of a publish-subscribe broker
core.ps.data	Topic-data resource of a publish-subscribe server

Table 2: CoAP Pubsub Resource Types

6.4. CoAP Pubsub Topic Properties Registry

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

The registration policy is either "Private Use", "Standards Action with Expert Review", or "Specification Required", or "Expert Review" per [BCP26]. "Expert Review" guidelines are provided in Section 6.5.

All assignments according to "Standards Action with Expert Review" are made on a "Standards Action" basis per Section 4.9 of RFC 8126 [BCP26] with "Expert Review" additionally required per Section 4.5 of RFC 8126 [BCP26]. The procedure for early IANA allocation of "standards track code points" defined in [BCP100] also applies. When such a procedure is used, IANA will ask the designated expert(s) to approve the early allocation before registration. In addition, working group chairs are encouraged to consult the expert(s) early during the process outlined in Section 3.1 of RFC 7120 [BCP100].

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 the CBOR key of the item. These values MUST be unique. The value is an integer (either positive or negative). Different ranges of values use different registration policies [BCP26]. Integer values from -256 to 255 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535 are designated as "Specification Required". Integer values greater than 65535 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.

This registry has been initially populated with the values in Table 1. Reference should always be [RFC-XXXX].

6.5. Expert Review Instructions

The registration policy for the IANA registry established in Section 6.4 is defined as one of "Standards Action with Expert Review", "Specification Required", and "Expert Review". This section gives some general guidelines for what the experts should be looking for; however, they are being designated as experts for a reason, so they should be given substantial latitude.

These registration policies are designed to accommodate different use cases; "Standards Action with Expert Review" allows for further IETF standards and extensions, maintaining consistency and alignment with established protocols; "Specification Required" allows third-party specifications from Standards Development Organizations (SDOs) to register topic properties, enabling interoperability and broader applicability; and "Expert Review" provides a flexible mechanism for exposing new properties that implementors do not want to keep in a private range.

Expert reviewers should take into consideration the following points:

* Clarity and correctness of registrations. Experts are expected to check the clarity of purpose and use of the requested entries. Experts need to make sure that registered topic properties are clearly defined in the corresponding specification. Properties that do not meet these objectives of clarity and completeness must not be registered.

* Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as "Private Use" are intended for testing purposes and closed environments. Code points in other ranges should not be assigned for testing.

* Specifications are required for the "Standards Action With Expert Review" range of point assignment. Specifications should exist for "Specification Required" ranges, but early assignment before a specification is available is considered to be permissible. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

* Experts should take into account the expected usage of fields when approving point assignment. Documents published via Standards Action can also register points outside the Standards Action range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

7. References

7.1. Normative References

[BCP100] Best Current Practice 100,
<https://www.rfc-editor.org/info/bcp100>.
At the time of writing, this BCP comprises the following:

Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[BCP13] Best Current Practice 13,
<https://www.rfc-editor.org/info/bcp13>.
At the time of writing, this BCP comprises the following:

Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, DOI 10.17487/RFC4289, December 2005, <https://www.rfc-editor.org/info/rfc4289>.

Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

D端rst, M.J., "Guidelines for the Definition of New Top-Level Media Types", BCP 13, RFC 9694, DOI 10.17487/RFC9694, March 2025, <https://www.rfc-editor.org/info/rfc9694>.

[BCP14] Best Current Practice 14,
<https://www.rfc-editor.org/info/bcp14>.
At the time of writing, this BCP comprises the following:

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/info/rfc2119>.

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/info/rfc8174>.

[BCP26] Best Current Practice 26,
<https://www.rfc-editor.org/info/bcp26>.
At the time of writing, this BCP comprises the following:

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/info/rfc8126>.

[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>.

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

[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>.

[RFC9176] Ams端ss, 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>.

[STD94] Internet Standard 94, <https://www.rfc-editor.org/info/std94>. At the time of writing, this STD comprises the following:

Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

7.2. 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-oscore-gm-admin] Tiloca, M., H旦glund, 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-15, 23 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-oscore-gm-admin-15>.

[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-11, 7 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-pubsub-profile-11>.

[I-D.ietf-core-interfaces] Shelby, Z., Koster, M., Groves, C., Zhu, J., and B. Silverajan, "Reusable Interface Definitions for Constrained RESTful Environments", Work in Progress, Internet-Draft, draft-ietf-core-interfaces-14, 11 March 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-core-interfaces-14>.

[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>.

[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>.

[RFC9594] Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication Using Authentication and Authorization for Constrained Environments (ACE)", RFC 9594, DOI 10.17487/RFC9594, September 2024, <https://www.rfc-editor.org/rfc/rfc9594>.

[STD96] Internet Standard 96, <https://www.rfc-editor.org/info/std96>. At the time of writing, this STD comprises the following:

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/info/rfc9052>.

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

Appendix A. Document Updates

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

A.1. Version -13 to -14

* Section restructuring for better readability.
* Updated topic configuration interactions.
* Introduced iPATCH section.
* Various clarifications of default values for parameters.
* New examples for several interactions.
* Updated topic discovery section.
* Other editorial changes

A.2. Version -14 to -15

* Code bug fix https://github.com/jaimejim/aiocoap-pubsub-broker/commit/f32ce4866a81319238d6e905de439c9410cce175
* Added two new optional topic configuration parameters; "initialize" and "topic-history".

* Modified all examples to conform to RFC9594.
* Added the explicit cancellation of ongoing subscriptions when topic configuration parameters are changed.
* Added editorial changes based on feedback.
* Clarifications on Topic Configuration creation.
* Other editorial changes

A.3. Version -15 to -16

* Various updates throughout the document based on AD review.
* IANA clarifications

A.4. Version -16 to -17

* Addressing Esko’s and Ari’s review.
* Fixing formatting

A.5. Version -17 to -18

* Addressed issues #64, #65, #66, #67.
* rt, ct, obs attribute elision
* Editorial changes

A.6. Version -18 to 19

* IANA early review
* Addressed issues #68, #69
* Addressed Marco’s review
* Addressed Marco’s and Christian Ams端ss’s WGLC reviews
* Redesigned "initialize" property for RFC7641 compliance
* Changed "expiration-date" to CBOR tag 1, CBOR keys numeric-only
* Relaxed overly strict response codes

* Clarified topic-data URI reference, immutable properties, and architecture roles

* Editorial fixes

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. The document follows a similar architectural design as that provided by Marco Tiloca’s [I-D.ietf-ace-oscore-gm-admin].

The authors would like to thank Marco Tiloca, Esko Dijk,Francesca Palombini, Christian Ams端ss, Carsten Bormann, Hannes Tschofenig, Zach Shelby, Mohit Sethi, Peter van der Stok, Tim Kellogg, Anders Eriksson, Goran Selander, Mikko Majanen, Olaf Bergmann, David Navarro, Oscar Novo and Lorenzo Corneo for their valuable contributions and reviews.

Contributors

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden
Email: marco.tiloca@ri.se

Marco offered comprehensive reviews and insightful guidance on the recent iterations of this document. His contributions were valuable un multiple parts of the document but particularly notable in the Security Considerations section.

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