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

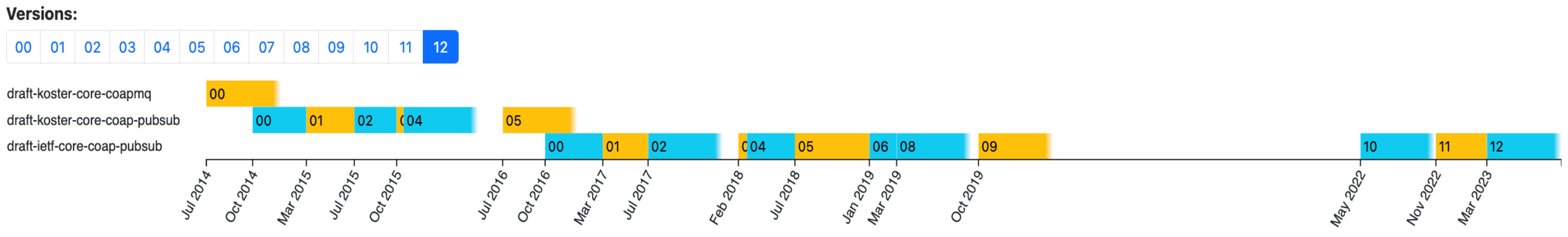
draft-ietf-core-coap-pubsub-12

Jaime Jiménez, Ericsson
Michael Koster
Ari Keränen, Ericsson

IETF 116 meeting - Yokohama - March 28th, 2023

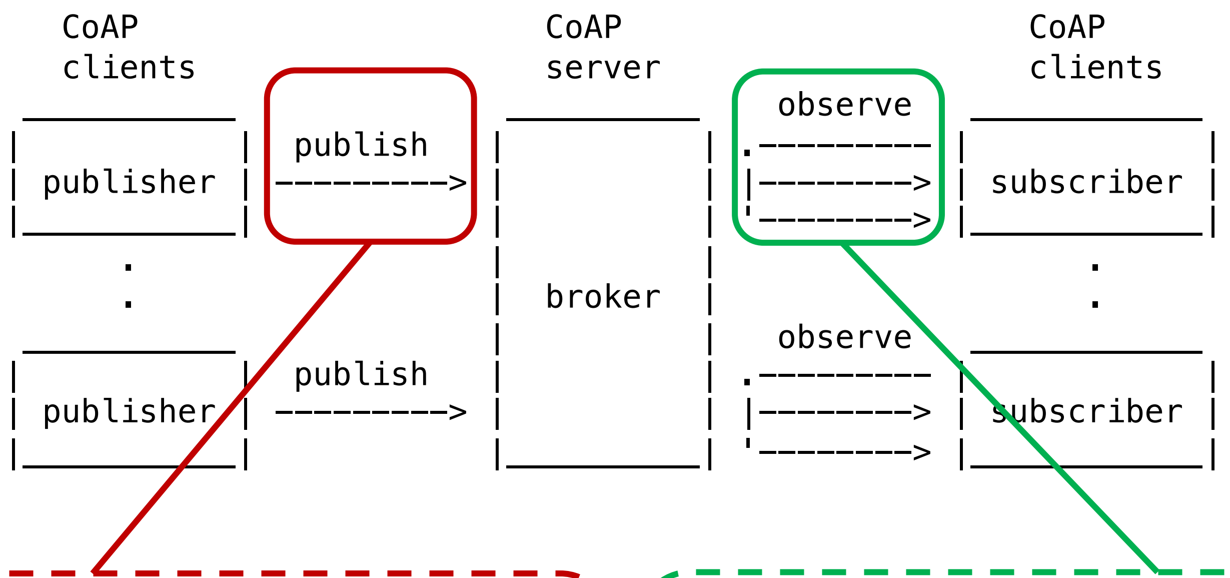
Draft History

- Together with *core-interfaces* and *core-dynlink* among the “senior” working group drafts we have in CoRE (2016).
- Current design is inspired by **hartke-t2trg-coral-pubsub** and **ietf-ace-oscore-gm-admin**



- This version (v12) introduces the topic configuration operations. The publish-subscribe over CoAP principle remain very similar.
- Easy to implement, very complete CoAP implementations out there nowadays.

Recap: publish-subscribe in CoAP

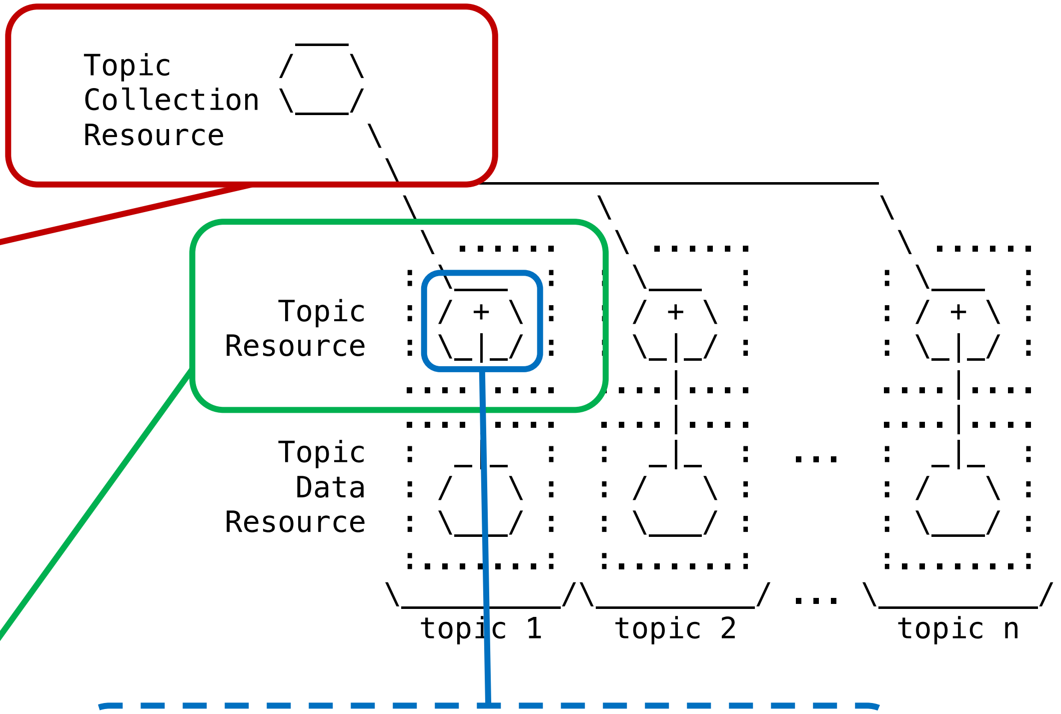


```
0.03 PUT <broker_URI>/ps/data/225acdd    =>

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

2.04 Changed                              <=
```

```
<= 0.01 GET <broker_URI>/ps/data/225acdd
Observe: 0

=> 2.05 Content Observe: 10001
[... Payload data...]

=> 2.05 Content Observe: 10002
[... Payload data...]

...
```

API Overview

Topic Collection resource
- Retrieve (GET) the list of topics
- Retrieve (FETCH) topics by properties
- Create (POST) a topic resource



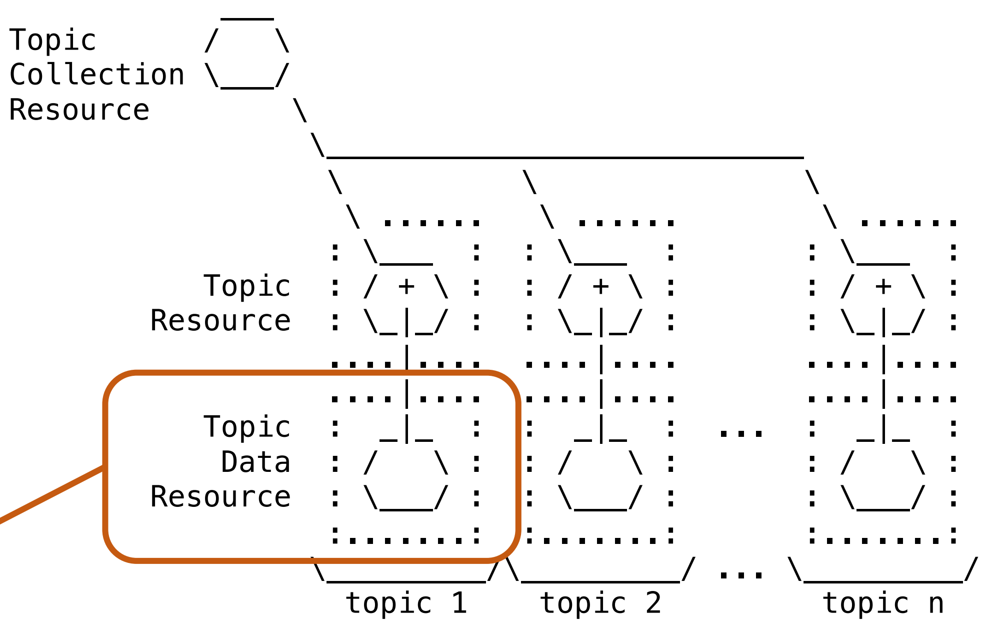
Topic resource (configuration)
- Retrieve (GET) a topic resource
- Retrieve (FETCH) part of a topic with a filter
- Update (PUT) whole topic
- Update (PATCH) part of a topic with a filter
- Delete (DELETE) a topic resource

Topic Properties
- Configuration parameters written by the administrator of the topic.
- Optional informational parameters (e.g., max_subscribers)

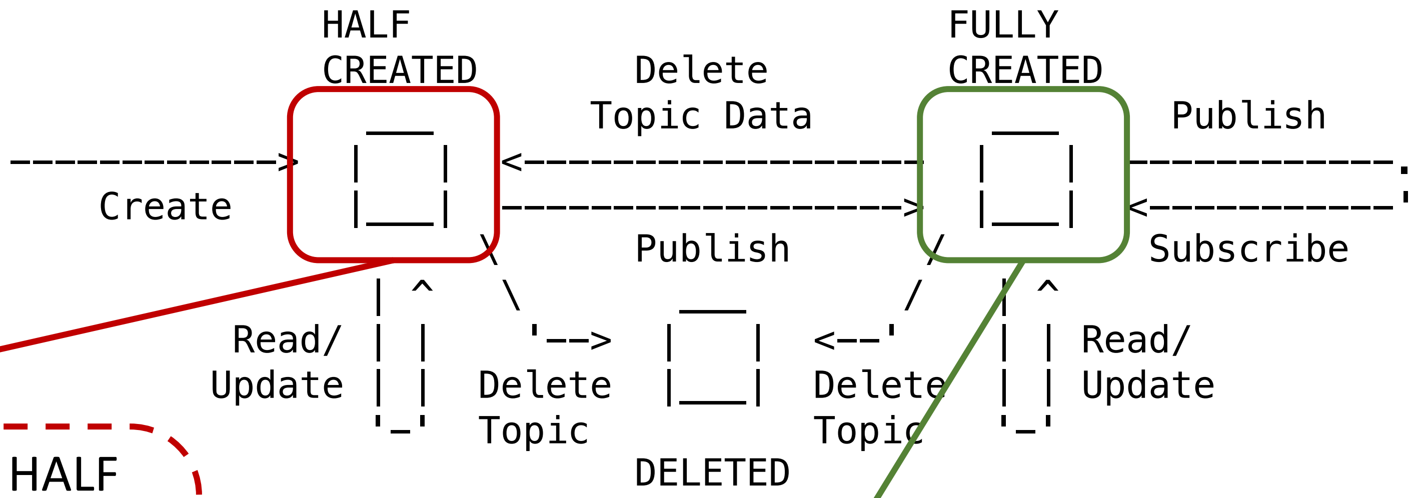
API Overview

Topic Data resource

- Publish (PUT) to a topic data (URI)
- Subscribe (GET + obs=0) to a topic data (URI)
- Unsubscribe (GET + obs=1) from a topic data (URI)
- Read latest value (GET)
- Delete (DELETE) a topic data



Topic Lifecycle



Topic configuration interactions, in the HALF CREATED state the topic is created but no data has been published to it.

```
=> POST /ps

{"topic_name": "Room Temperature Sensor",
"resource_type": "core.ps.conf", "media_type":
"application/json", "target_attribute":
"temperature", "expiration_date": "2023-04-
05T23:59:59Z", "max_subscribers": 100}

<= 2.01 Created
location: ps/7b7275

{"topic_name": "Room Temperature Sensor",
"topic_data": "ps/data/55741fd", "resource_type":
"core.ps.conf"}
```

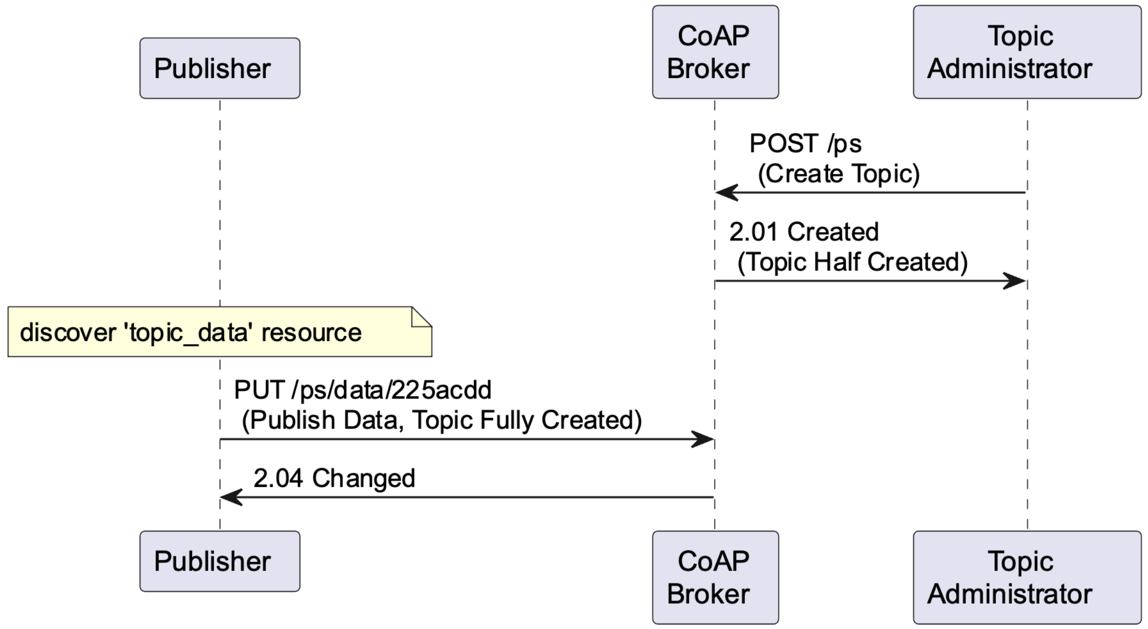
A publisher publishes on the topic data resource **ps/data/55741fd**

```
=> PUT /ps/data/55741fd

[{"n": "temperature","u":
"Cel","t": 1621452122,"v":
21.3}]
```

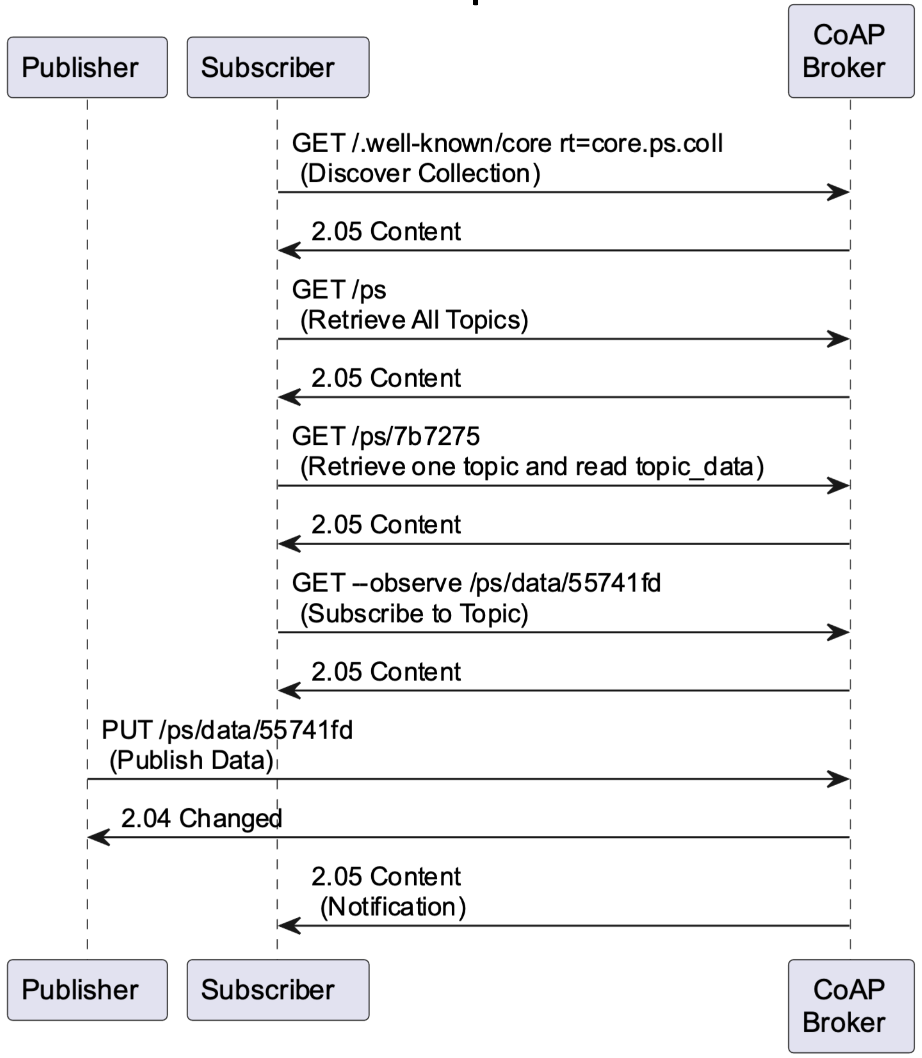
The state changes to FULLY CREATED. Subscribers can now subscribe and publish on that resource.

Workflow Example

Create a Topic


Publisher
CoAP Broker
Topic Administrator
POST /ps (Create Topic)
2.01 Created (Topic Half Created)
discover 'topic_data' resource
PUT /ps/data/225acdd (Publish Data, Topic Fully Created)
2.04 Changed
Publisher
CoAP Broker
Topic Administrator

Interact with a Topic


Publisher
Subscriber
CoAP Broker
GET /.well-known/core rt=core.ps.coll (Discover Collection)
2.05 Content
GET /ps (Retrieve All Topics)
2.05 Content
GET /ps/7b7275 (Retrieve one topic and read topic_data)
2.05 Content
GET --observe /ps/data/55741fd (Subscribe to Topic)
2.05 Content
PUT /ps/data/55741fd (Publish Data)
2.04 Changed
2.05 Content (Notification)
Publisher
Subscriber
CoAP Broker

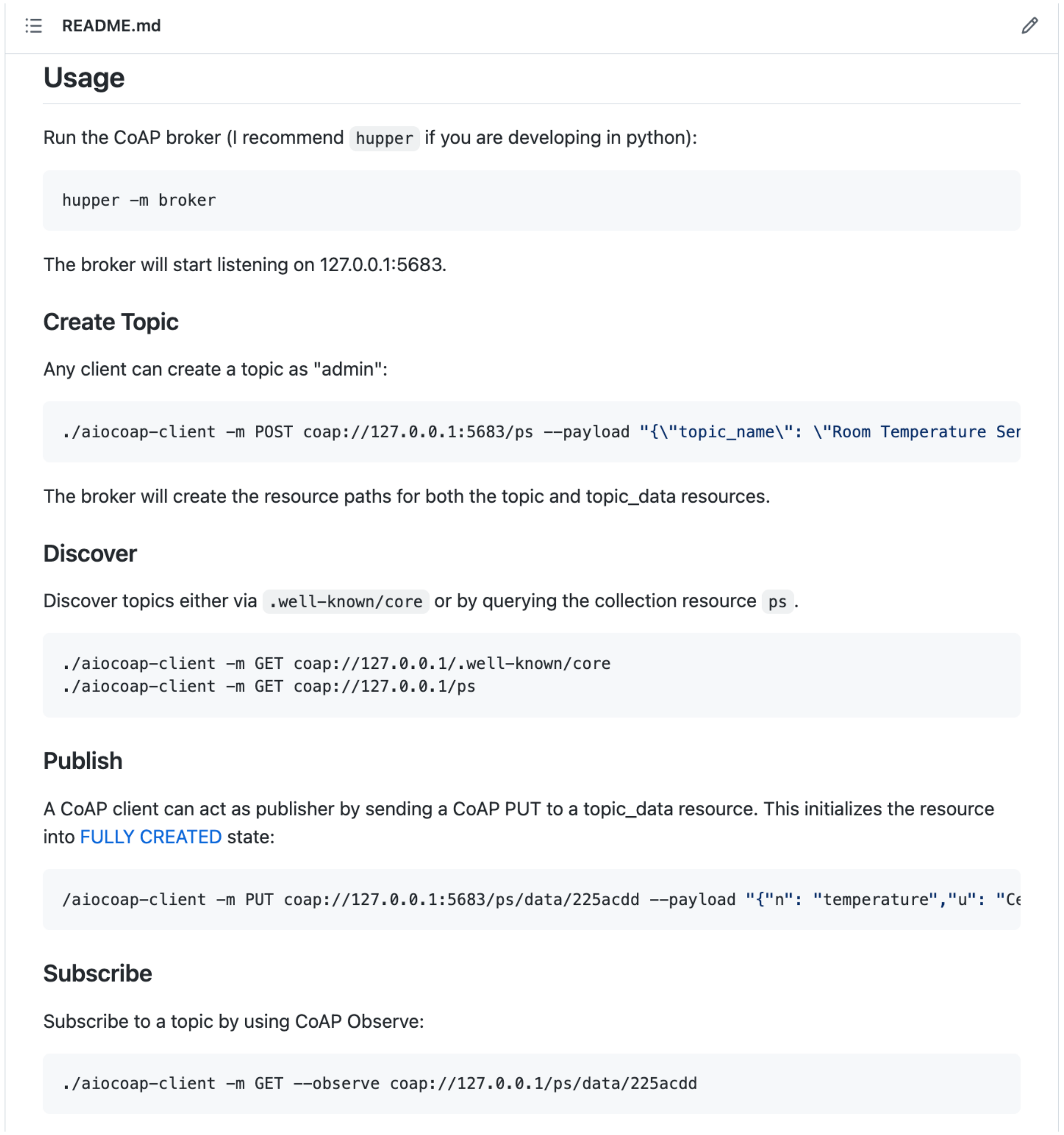
Hackathon Implementation

github.com/jaimejim/aiocoap-pubsub-broker

A simple python implementation of the topic discovery, configuration and pub-sub topic data interactions on top of aiocoap.

The broker implements the following resource classes:

- CollectionResource: The collection resource /ps for storing topics.
- TopicResource: A resource for topic configurations.
- TopicDataResource: A resource for topic data and for the publish-subscribe interactions over CoAP.



README.md

Usage

Run the CoAP broker (I recommend `hupper` if you are developing in python):

```
hupper -m broker
```

The broker will start listening on 127.0.0.1:5683.

Create Topic

Any client can create a topic as "admin":

```
./aiocoap-client -m POST coap://127.0.0.1:5683/ps --payload "{\"topic_name\": \"Room Temperature Ser
```

The broker will create the resource paths for both the topic and topic_data resources.

Discover

Discover topics either via `.well-known/core` or by querying the collection resource `ps`.

```
./aiocoap-client -m GET coap://127.0.0.1/.well-known/core
./aiocoap-client -m GET coap://127.0.0.1/ps
```

Publish

A CoAP client can act as publisher by sending a CoAP PUT to a topic_data resource. This initializes the resource into FULLY CREATED state:

```
/aiocoap-client -m PUT coap://127.0.0.1:5683/ps/data/225acdd --payload "{"n": "temperature","u": "Ce
```

Subscribe

Subscribe to a topic by using CoAP Observe:

```
./aiocoap-client -m GET --observe coap://127.0.0.1/ps/data/225acdd
```

Discussion

- Are there some topic properties missing or underspecified?
- ‘topic_name’ is an application identifier, do we want to define some UUID/URN space for it? Maybe not? Right now this is not a field that the broker can autogenerate, is that OK?
  - CB: within a single collection of topic resources it should be unique.
- Do we want to treat ‘max_subscribers’ as an error? Now we use RFC7641: *The resulting (2.05) response MUST NOT include an Observe Option.*
  - MT: do we want ‘max_clients’ field for subscribers+publishers
- Authorization for admin operations are out of scope, are there some parts of it that really should be included or are we OK with that? “topic creator/subscriber privileges”?
- Security is already enabled by CoAP Ecosystem (CoAP + oscore + dtls). Security Consideration section is temporarily needs some coordination with ACE:
  - Draft is intended to work with different security models.
  - ACE draft-ietf-ace-pubsub-profile covering authorization for users.
  - Topic Creation/Discovery requirements (topic manager approval).

Next Steps for v13

- Topic configuration and data resources can be hosted on different servers, reflect that on the draft.
- IANA section
- Use all of max-age, etc, correctly.
- **Security section + references to ACE draft**
- Use CBOR on the implementation too, implement missing operations.