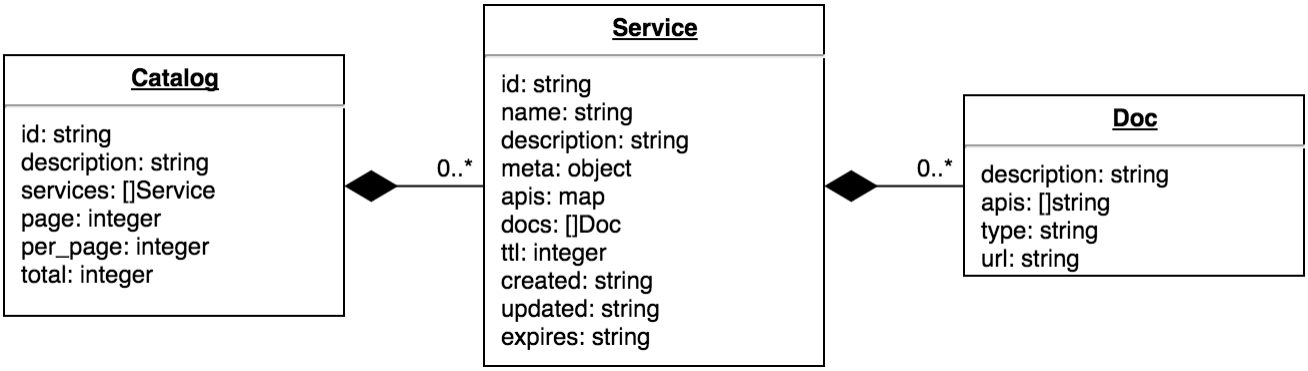
Service Catalog API

Overview

Service Catalog provides a REST API for service discovery described below:

The following diagram shows the data model of Service Catalog:



The attributes are described below:

Catalog object consists of:

- id: unique id of the catalog
- description: a friendly name or description of the service
- services: an array of Service objects
- page: the current page in catalog
- per_page: number of items in each page
- total: total number of registered services

Service object consists of:

- id: unique id of service
- name: RFC6339 service name (e.g. _bms._tcp)
- description: friendly name or description of a service
- meta: a hash-map for optional meta-information
- apis: a map of API names and URLs
- docs: an array of Doc objects describing service documentations
- ttl: time after which the service should be removed from the catalog, unless if it is updated within the timeframe.
- created: RFC3339 time of service creation
- updated: RFC3339 time in which the service was lastly updated
- expires: RFC3339 time in which the service expires and is removed from the catalog (only if TTL is set)

Doc object consists of:

- description: description of the external document
- apis: an array listing APIs documented in this documentation
- url: URL to the external document
- type: the MIME type of the document (e.g. plain/text for wikis, application/openapi+json;version=2.0 for OpenAPI specs v2.0)

MQTT API

Service catalog also supports MQTT for service registration and de-registration.

Service registration is similar to POST and PUT methods of REST API. Here, a service uses a pre-configured topic (via config file) for publishing the message.

The will message of the registered service is used to de-register it from the cataog whenever the service disconnects.

Announcement over MQTT:

Service Catalog Announces the service registration status via MQTT using retain messages.

The the message topics follow following patterns:

<topicPrefix >/< RFC6339 service name>/<service_id>/alive : (Retained message) The body contains service description of alive service

<topicPrefix >/< RFC6339 service name>/<service_id>/dead : (Not retained message) The body contains service description of alive service

The retained messages are removed whenever service de-registers.

topicPrefix can be configured via config file.

Versioning

API version is based on semver. The version is included as a parameter to the MIME type of all HTTP responses:

```
application/json;version=X.X.X
```