decode

D3.2 Data access and transaction management module

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

D.3.2 Data access and transaction management module

Version Number: 1.0.1

Lead beneficiary: Thingful

Due Date: August 2017

Author(s): Mark deVilliers (Thingful)

Editors and reviewers: Jim Barritt (Thoughtworks) and Japp-Henk Hoepman (RU)

Dissemination level:		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: Francesca Bria (Chief Technology and Digital Innovation Officer, Barcelona City Hall)

Date: 28/08/2107

This report is currently awaiting approval from the EC and cannot be not considered to be a final version.

Contents

Introduction

The proposal describes the deliverable for 3.2 as :

D3.2 Data access and transaction management module (M8)

An early stage prototype working demonstrator which is capable of: private IoT data search, entitlement match, decentralized data access and transaction enablement.

The deliverable for 3.2 takes the form of a software repository.

The address for the repository is - https://gogs.dyne.org/DECODE/decode-prototype-da

About the software

The following is adapted from the file https://gogs.dyne.org/DECODE/decode-prototype-da/src/master/README.md

This repository contains an early stage prototype working demonstrator which is capable of:

- retaining private IoT data

- search of private IOT data

- demonstrating an implementation of data entitlement specifically around the visibility and access of private IoT data

- decentralized data access of private IoT data

The architectural context for the prototype is based around the discussions in the first 6 months of the DECODE project (https://decodeproject.eu/) by the technical partners including Thoughtworks (https://www.thoughtworks.com/), Dyne (https://www.dyne.org/), UCL(http://sec.cs.ucl.ac.uk/home/) and thingful (https://www.thingful.net).

The purpose of the prototype is to demonstrate some of the concepts from the DECODE project grounded in an IoT use-case.

Data Entitlement

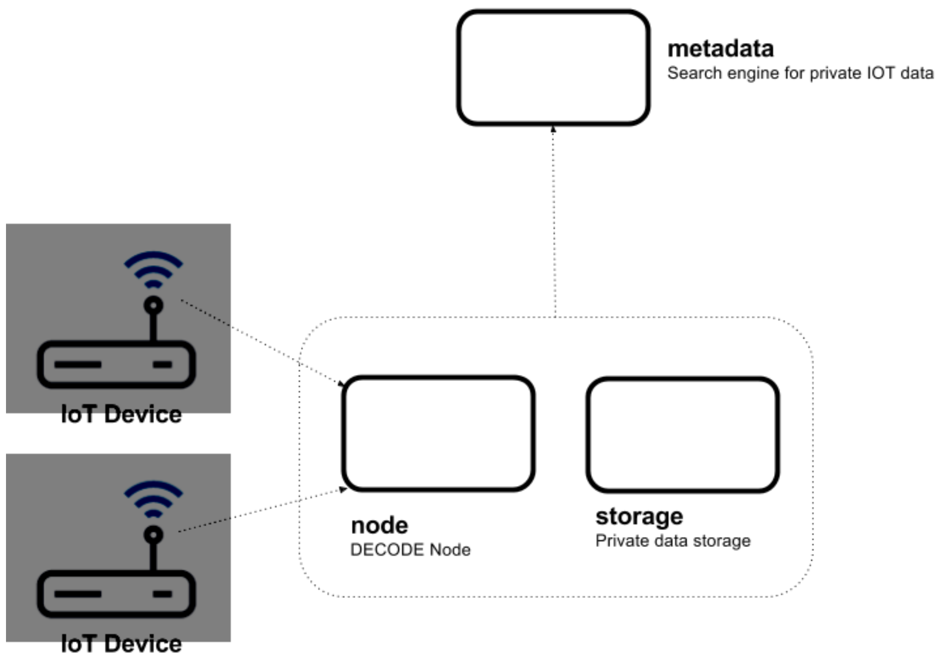
An introduction to concepts of data entitlements as well as a review of prior art was completed as part of the DECODE project.

From the review:

Data entitlements could be thought of as an evolution of a traditional authorization scheme specialized for the securing of both personal, business and IoT data. Giving the data owner full control of the access and discovery of their data creates a system of empowerment whose currency is privacy....The relationship of privacy and data entitlement is subject to many nuances. Users might allow a party to search and access their data, but only under the understanding that they will not be personally identified or only under specific circumstances. For example, a driver of a motor vehicle with an on-board camera might want to entitle the emergency services access to his data when there is a road traffic accident. However, the driver might not want to be identified as being in a certain geographical area or that he was exceeding the speed limit at the time. Data,when available, can directly or indirectly compromise privacy in a way that would surprise a user.

The full document, D3.1 Survey of Technologies for ABC, Entitlements and Blockchains, can be found here - https://desk.dyne.org/s/7gHFaUIoNEFB1Wx#pdfviewer.

For the purposes of the prototype any singular piece of data in our system will have a data entitlement in one of three logical states -



- **owner-only** - the data is not discoverable or accessible and is essentially private.

- **can-discover** - the data is discoverable and will be made available for search. The data is not accessible.

- **can-access** - the data is accessible and by default discoverable.

To qualify the terms used above :

- **discoverable** – the existence of the data can be found.

- **accessible** – the value of the data can be viewed. For the data to be accessible it is of course discoverable.

It is important to note that the prototype does not, by design, implement authentication and as such all entitlements once given are applied globally.

Architecture

The prototype is implemented as a set of HTTP services, written in golang, exposing various endpoints accepting various JSON messages in a REST-like manner.

Each of the services is documented with a swagger api definition. More details can be found at https://swagger.io/.

An overview of a single node system.

In this diagram two IoT devices logically direct their data to the *node* service.

As a convenience we have included some “fake” IoT devices as part of the prototype.

The output of these devices correspond to the output of the device-hub (https://github.com/thingful/device-hub) component as developed by thingful (https://thingful.net) and open-sourced as part of thingful's involvement in the DECODE project.

The *node* service is the entry point for an instance of the storage service.

The *node* has a relationship with the *metadata* service.

The *node* service is responsible for:

- maintaining the configuration of data entitlements.

- accepting data from one or more sources - in the prototype this is a selection of IOT devices.

- orchestrating the discover-ability and accessibility of the data as per the configured data entitlements.

The *node* service has a *storage* service which is responsible for:

- maintaining any saved data points

The *node* service represents the concept of the DECODE node in the DECODE Architecture white-paper. Please note this has not been made public yet.

The *metadata* service is responsible for:

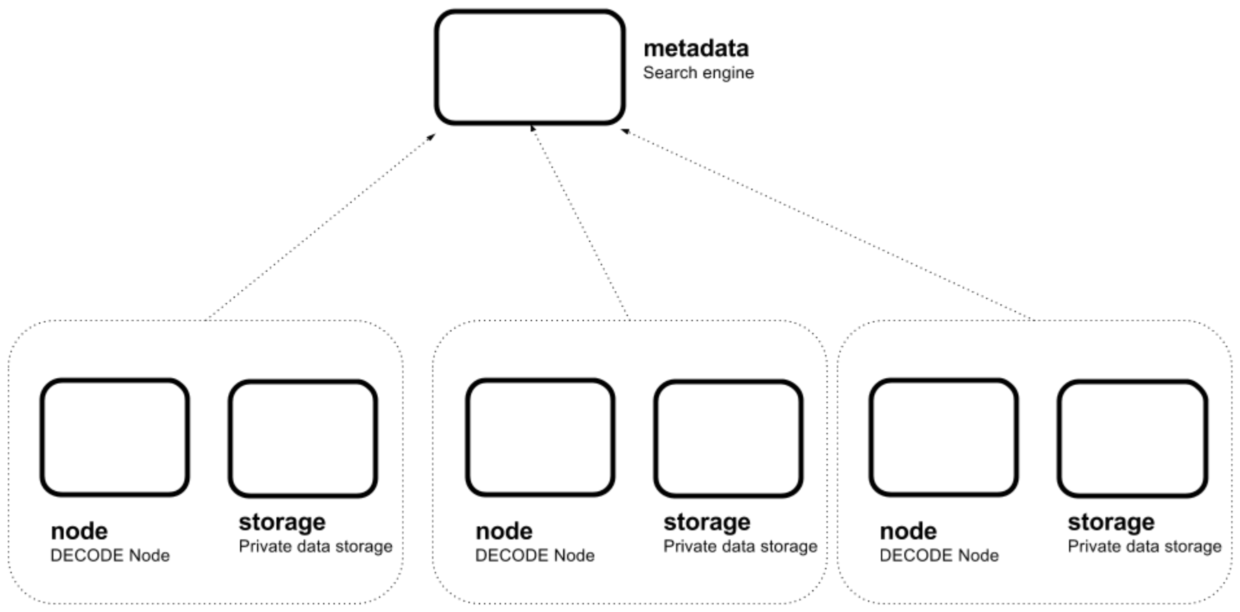
- maintaining an index of discoverable data and their locations.

- displaying any data that is accessible.

- allow a user to make an entitlement request to access data that is not yet accessible.

Both the *node* and *metadata* service expose a user interface.

The architecture is designed to scale with multiple DECODE nodes using a single metadata service.



The above diagram shows three *node* instances and a single global *metadata* instance.

Ontological mapping

Data from the *fake* devices have been tagged with the following ontological data.

Device	Value	Tag	Notes
Fake sine wave sensor	Value	http://decode.eu#Fun	sine wave data is fun. Please note that this ontology is made up for the purpose of this prototype.
		http://www.w3.org/2001/XMLSchema#float	sine wave data is a float value
		http://purl.org/iot/vocab/m3-lite#Sensor	sine wave data is sensor data
Fake temperature humidity sensor	Temperature	http://purl.org/iot/vocab/m3-lite#AirTemperature	Temperature is AirTemperature
		http://purl.org/iot/vocab/m3-lite#Environment	Temperature is environmental data
		http://purl.org/iot/vocab/m3-lite#Sensor	Temperature is sensor data
		http://www.w3.org/2001/XMLSchema#float	Temperature is a float value
	Humidity	http://purl.org/iot/vocab/m3-lite#AirHumidity	Humidity is AirHumidity
		http://purl.org/iot/vocab/m3-lite#Environment	Humidity is environmental data
		http://purl.org/iot/vocab/m3-lite#Sensor	Humidity is sensor data
		http://www.w3.org/2001/XMLSchema#float	Humidity is a float value

The strategy applied for the selection of schemas is to reuse existing schemas where possible and to fill any gaps with a proposed *DECODE* schema.

The m3-lite ontology (http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot/doc) was developed as part of the EU 2020 Fiesta-IoT project and is used extensively at thingful.

Notes

- this is a prototype and not production software.

- there is no authentication and or authorization.

- all data is held in memory - resetting the environment will reset all of the data.

Building and Running

To build the software ensure you have installed the following software :

- Golang 1.7.3+

- Docker and Docker compose

- Elm 0.18+

Once you have a working installation download the code using the go get

```
go get gogs.dyne.org/DECODE/decode-prototype-da
```

The makefile contains helpers to build the environment plus some helpers for development.

```
make help
```

To build all of the docker containers locally for docker compose to use

```
make docker-build
```

To run the application components via docker compose

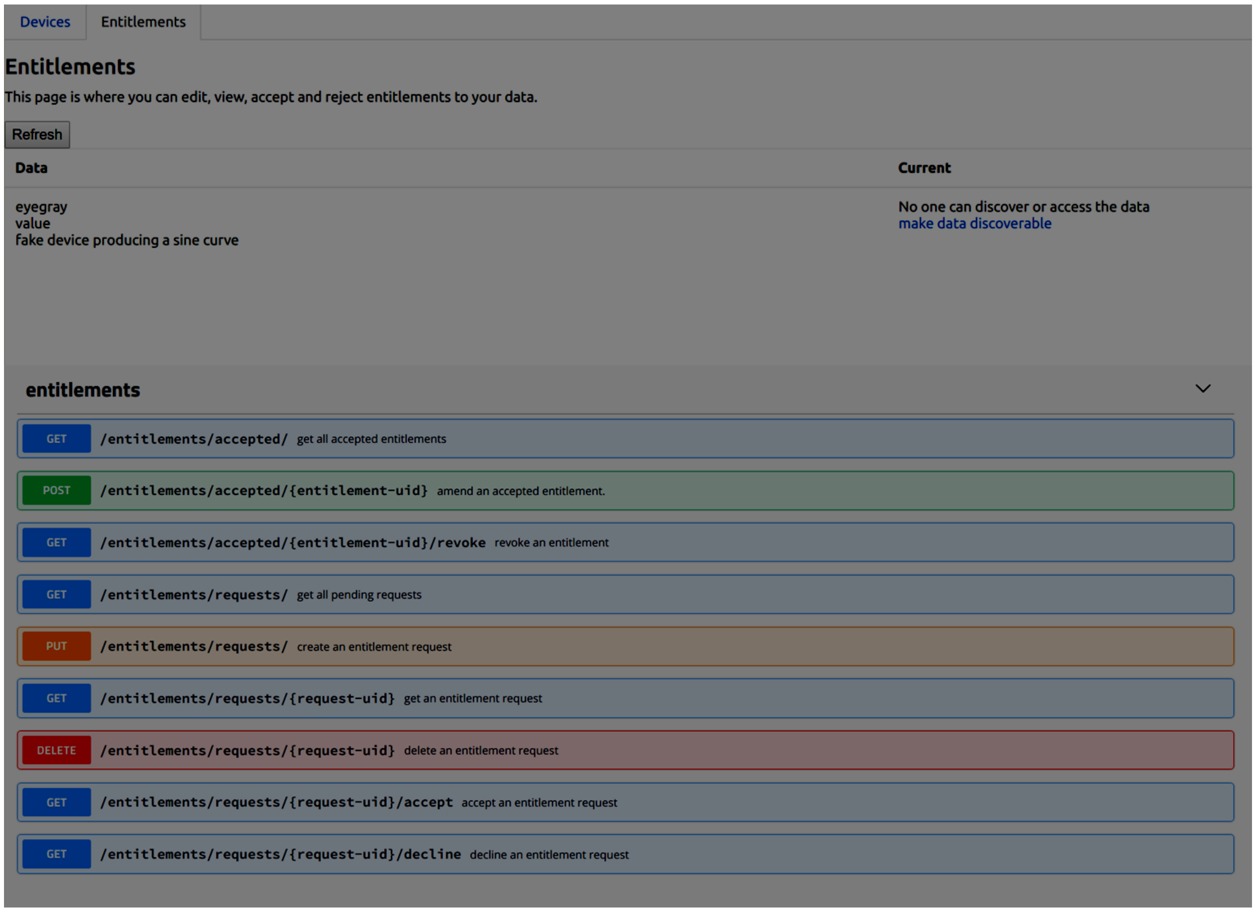
```
docker-compose up
```

The docker-compose tool will then map the ports exposed by the containers to local ports on you own machine.

The ports it will try and use are - 8080, 8081 and 8083.

If theses ports are already in use the docker-compose tool will display a warning.

The prototype should then be available at the following urls


Devices | Entitlements

Entitlements

This page is where you can edit, view, accept and reject entitlements to your data.

Refresh

Data	Current
eyegray value fake device producing a sine curve	No one can discover or access the data make data discoverable

entitlements

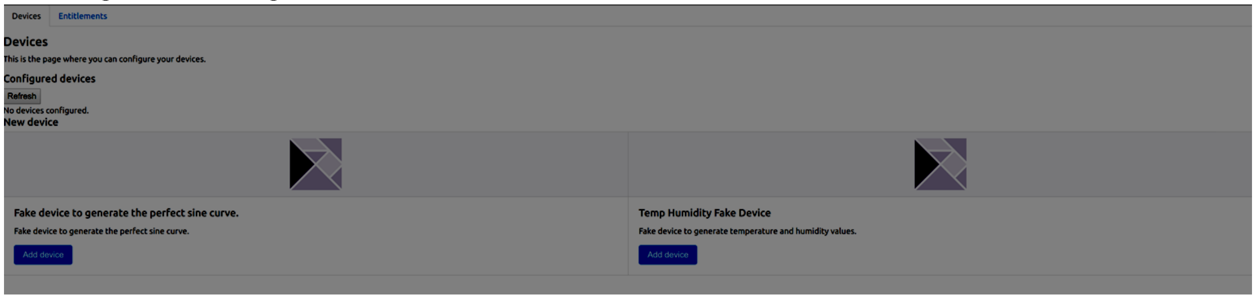
```
GET     /entitlements/accepted/  get all accepted entitlements
POST    /entitlements/accepted/{entitlement-uid}  amend an accepted entitlement.
GET     /entitlements/accepted/{entitlement-uid}/revoke  revoke an entitlement
GET     /entitlements/requests/  get all pending requests
PUT     /entitlements/requests/  create an entitlement request
GET     /entitlements/requests/{request-uid}  get an entitlement request
DELETE  /entitlements/requests/{request-uid}  delete an entitlement request
GET     /entitlements/requests/{request-uid}/accept  accept an entitlement request
GET     /entitlements/requests/{request-uid}/decline  decline an entitlement request
```

The *node* swagger api - http://localhost:8080/apidocs

The *node* UI- http://localhost:8080

The node UI allows for:

- adding and viewing IoT devices


Devices | Entitlements

Devices

This is the page where you can configure your devices.

Configured devices

Refresh

No devices configured.

New device

Fake device to generate the perfect sine curve.

Fake device to generate the perfect sine curve.

Add device

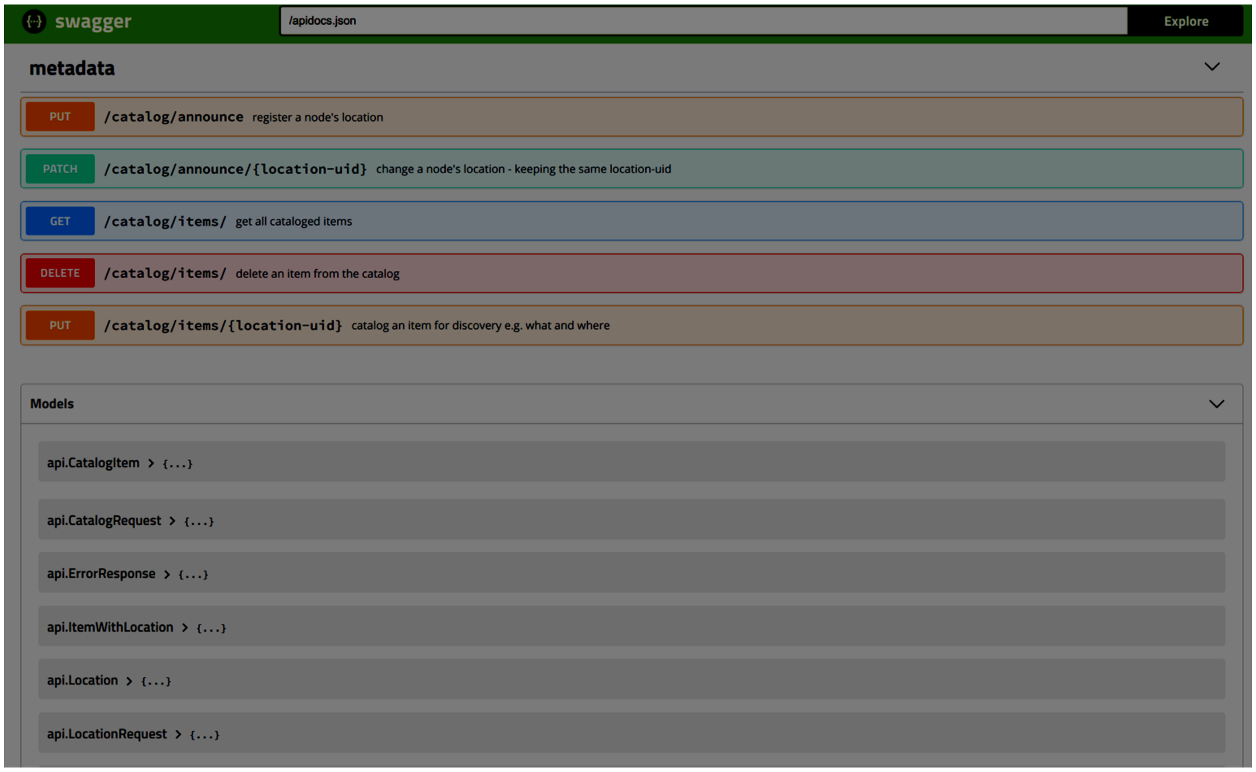
Temp Humidity Fake Device

Fake device to generate temperature and humidity values.

Add device

- viewing and editing data entitlements

The *metadata* swagger api - http://localhost:8081/apidocs


swagger /apidocs.json Explore

metadata

PUT /catalog/announce register a node's location

PATCH /catalog/announce/{location-uid} change a node's location - keeping the same location-uid

GET /catalog/items/ get all cataloged items

DELETE /catalog/items/ delete an item from the catalog

PUT /catalog/items/{location-uid} catalog an item for discovery e.g. what and where

Models

api.CatalogItem > {...}

api.CatalogRequest > {...}

api.ErrorResponse > {...}

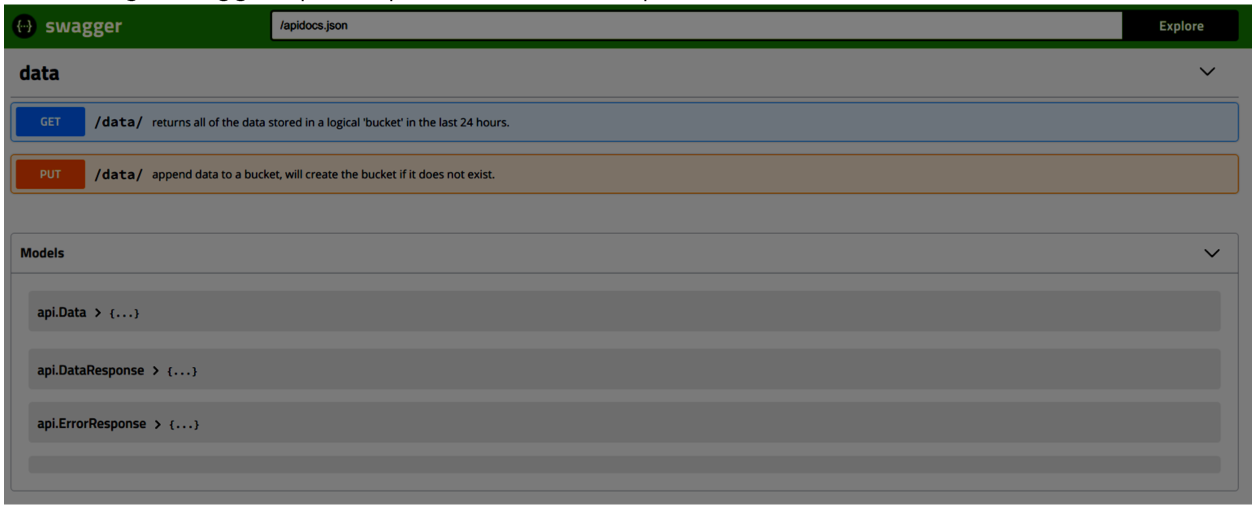
api.ItemWithLocation > {...}

api.Location > {...}

api.LocationRequest > {...}

The *metadata* ui - http://localhost:8081

The *storage* swagger api - http://localhost:8083/apidocs


swagger /apidocs.json Explore

data

GET /data/ returns all of the data stored in a logical 'bucket' in the last 24 hours.

PUT /data/ append data to a bucket, will create the bucket if it does not exist.

Models

api.Data > {...}

api.DataResponse > {...}

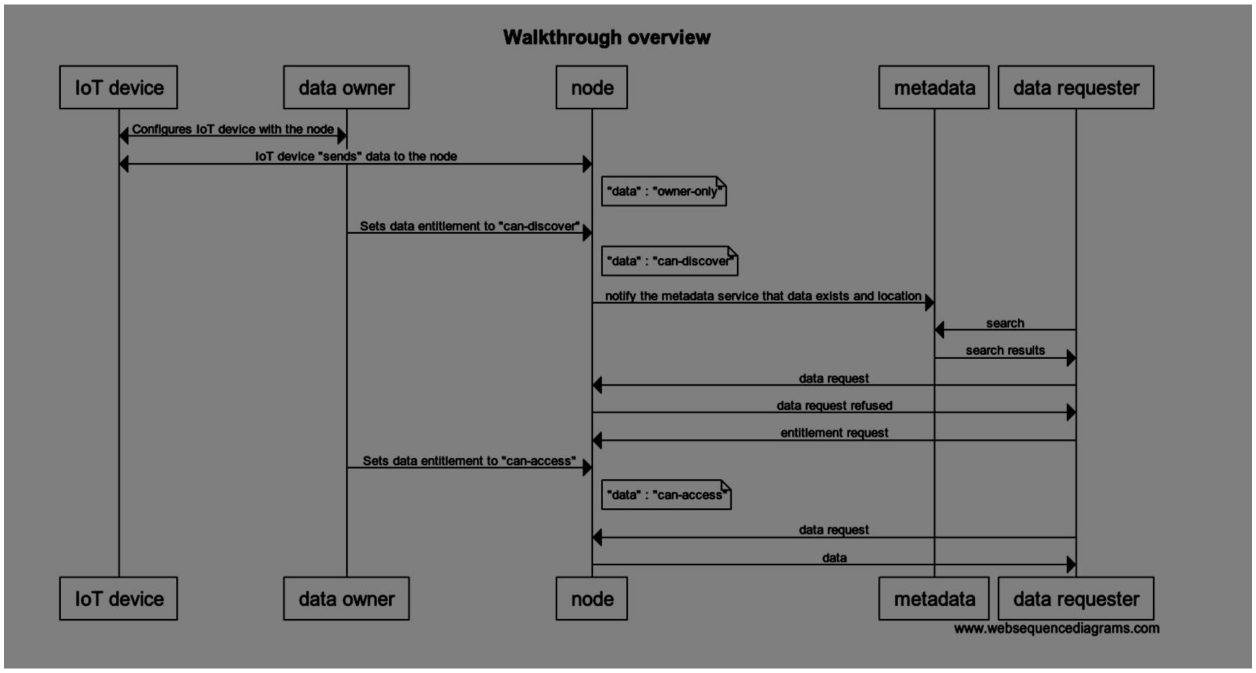
api.ErrorResponse > {...}

To stop the application components via docker compose you can either type Ctrl-C or from another window issue the command

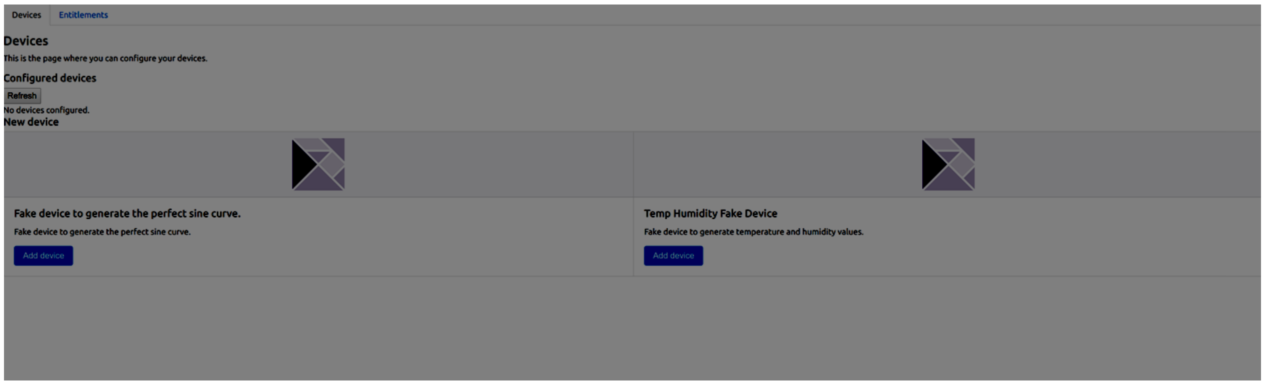
docker-compose down

Walk through

This walk through will demonstrate using the prototype from both a data owner and a data requesters perspective.


Walkthrough overview
IoT device
data owner
node
metadata
data requester
Configures IoT device with the node
IoT device "sends" data to the node
"data" : "owner-only"
Sets data entitlement to "can-discover"
"data" : "can-discover"
notify the metadata service that data exists and location
search
search results
data request
data request refused
entitlement request
Sets data entitlement to "can-access"
"data" : "can-access"
data request
data
www.websequencediagrams.com

The *node* application shows that no IoT devices have been configured.



Devices

This is the page where you can configure your devices.

Configured devices

No devices configured.

New device

Fake device to generate the perfect sine curve.

Fake device to generate the perfect sine curve.

Temp Humidity Fake Device

Fake device to generate temperature and humidity values.

The data owner adds a "Fake device to generate the perfect sine curve"

no data available for search.

The device is given the name “wanderpolar” and is listed as a configured device.

On the “entitlements” tab we can see that a data entitlement is available to be configured. By default the data is private.

This is confirmed by the UI in the *metadata* service. There is only one node and all of its data is private.

Clicking the “make data discoverable” link will now send the metadata for the item to the *metadata* service.

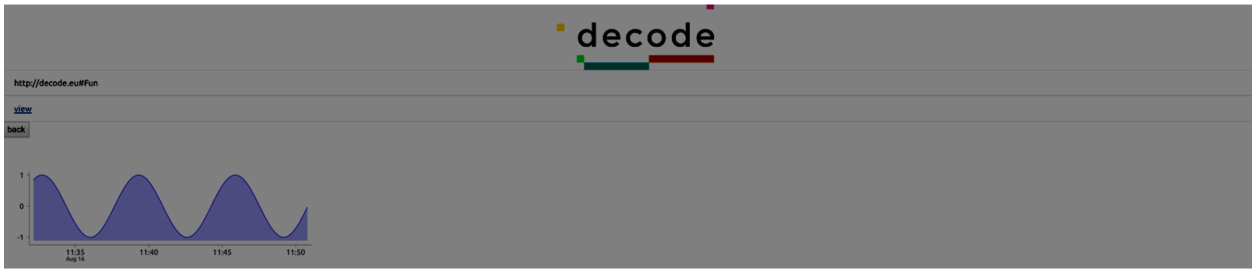
The *metadata* screen now shows that we have three items of metadata now available. Our sine wave data is tagged to three values

Tag	Notes
http://decode.eu#Fun	sine wave data is fun
http://www.w3.org/2001/XMLSchema#float	sine wave data is a float value
http://purl.org/iot/vocab/m3-lite#Sensor	sine wave data is sensor data

Selecting the first option shows us that there is one piece of data tagged to this value.

Selecting view asks the data requester if they would like to “request access” to the value. It is important to note that the *metadata* service does not know what the data entitlement for the data is. The *node* service has refused access to the data.

The data requester selects the “request access” link.



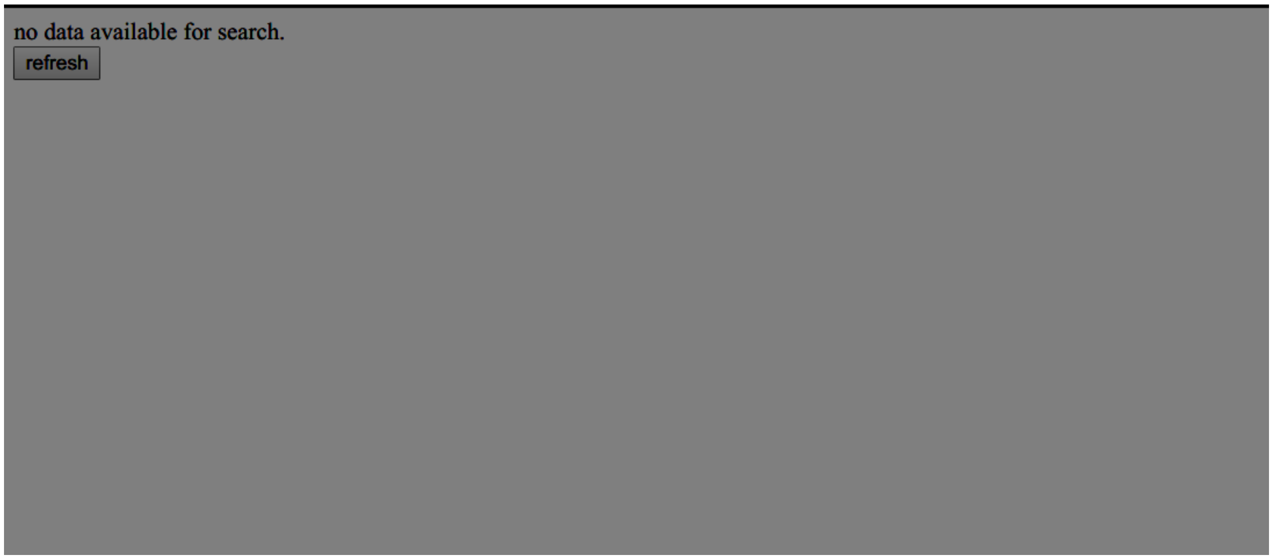
In the node service we can now see the request for access. The data owner can accept or decline he request. Declining the request makes no change to the data entitlement. Accepting the request will make the data viewable.

The data owner accepts the request and the entitlement is updated accordingly.

The data requester can now attempt to view the data again. The data is displayed – the perfect sine graph.

At some point in the future the data owner decides to make their data private again.

Their metadata is removed for the *metadata* service.


no data available for search.