TagItSmart
Smart tags driven service platform for enabling ecosystems of connected objects

Grant agreement 688061

SO Engine
IoT interoperability module

Deliverable ID:	**D1.2**
Deliverable Title:	Documentation for the IoT interoperability module
Project acronym	**SO Engine**
Project title:	**Smart Offers Engine for retail**
Extension acronym:	
Extension title:	
Contractual date of delivery:	**3/1/2018**

Contents

Introduction ***3***

Interoperability between TIS and SO Engine ***3***

1.1 Products and things **3**

1.2 Interoperability module **3**

1.3 Methods and API **4**

1.3.1 THNG creation 4

1.3.2 Redirection 5

1.3.3 Tag reading 6

Conclusion ***6***

Figures

Figure 1 - Inteoperability TIS/So Engine 4

Introduction

This report describes the interfaces between the TagItSmart platform and SO Engine.
We used the APIs developed by Evrythng.

Interoperability between TIS and SO Engine

1.1 Products and things

Product resources[1] are used to model a class of objects, and as such should contain information common to all instances. For example, a product resource could be used to describe a particular model of a physical product with specific characteristics (perhaps to match a specific SKU).
A ***Thng***[2] is the central data structure in the EVRYTHNG Platform. Thngs are extensible data containers designed to model physical objects such as packaged products, electronic devices, consumer electronics, buildings and spaces, etc. Thngs can be the target of actions describing events that happen to them, or they can be added to collections to express logical groupings, such as a box of individual items, as a unit. In addition to specifying a Thng in an action, it is also possible to create an action through the Thng itself using aliased actions.
So a thing (Thng) is an instance of a product category.

For performance and privacy reasons (no shared product and price databases for the retailer, API call with some latency for product creation[3]), the databases for things and products are internal to the SO Engine platform (in a compatible format), but if needed (common product passport) we could use the TIS Product and Thng databases

1.2 Interoperability module

The SO Engine platform has been developed in Python and Flask, and databases with MySQL. We have developed a completely specific version of SO Engine for this project. In the following, we will use the terminology Thng for a TIS thing and Thing for a SO thing.

Figure 1 summarizes the interactions between TIS and SO Engine.

The main interactions are :

- **Thng creation** : each thing has an Id, and is linked to a product category
- **Tag redirection** is used for the tag generation or on an change event for a Thng (usually, redirection is not needed, unless a new tag is required for a major change)
- **Tag generation** provides a tag url which is linked to the Thing (SO database)
- **Tag reading** is used in the SO mobile app

[1] https://developers.evrythng.com/v3.0/reference#products
[2] https://developers.evrythng.com/v3.0/reference#thngs
[3] It could be interesting to add an API for “batch” creation of products from a csv to reduce the lantency

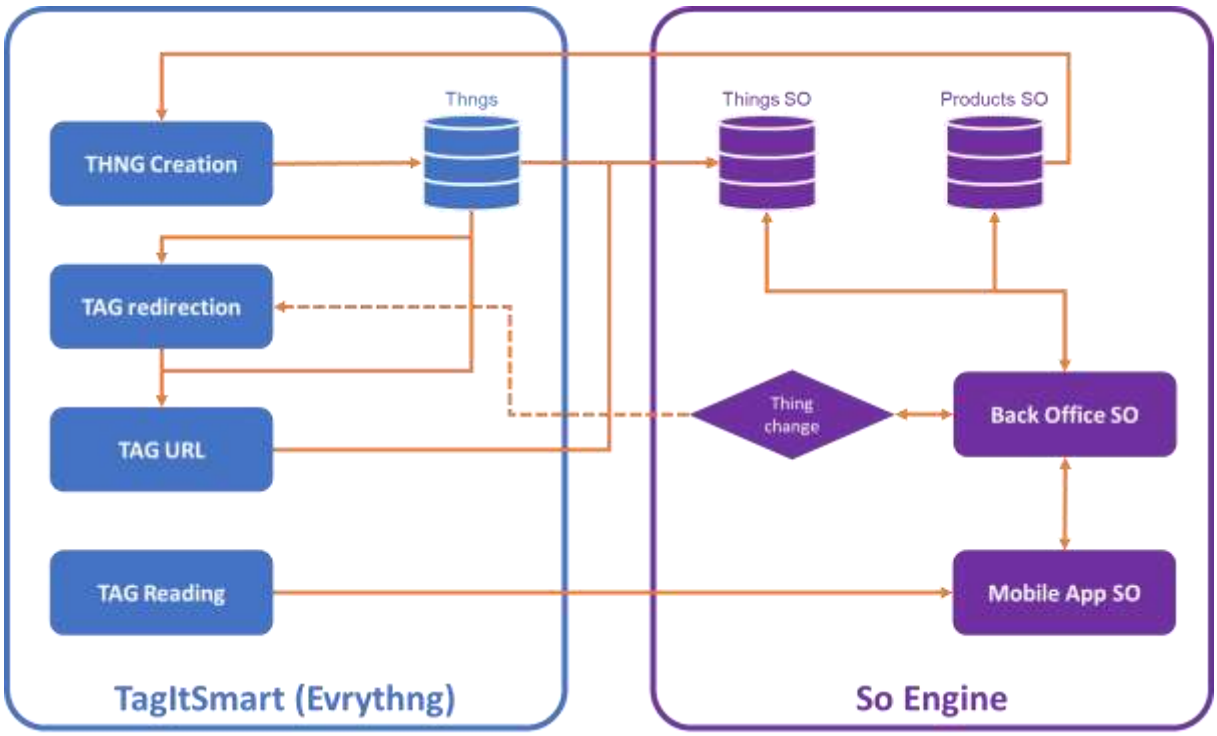

Figure 1 - Inteoperability TIS/So Engine

1.3 Methods and API

We describe here the TIS/Evrythng API used in the project.

1.3.1 THNG creation[4]

We use *[POST] /thng.*

Example :

```
curl -i -H "Content-Type: application/json" \
  -H "Authorization: $OPERATOR_API_KEY" \
  -X POST "https://api.evrythng.com/thngs" \
  -d '{
    "name": "Truck # L4-4",
    "description": "Truck L4-4. Delivery truck for North England.",
    "location": {
      "position": {
        "type": "Point",
        "coordinates": [ 2.05, 57.1 ]
```

[4] https://developers.evrythng.com/v3.0/reference#section-create-a-thng

```
      }
    },
    "properties": {
      "status": "Idle"
    }
  }'
```

Which returns :

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: https://api.evrythng.com/thngs/UFhB5TENG6ffeHxUCRt8eNta

{
  "id": "UFhB5TENG6ffeHxUCRt8eNta",
  "createdAt": 1346856615704,
  "updatedAt": 1346856615704,
  "name": "Truck # L4-4",
  "description": "Truck L4-4. Delivery truck for North England.",
  "location": {
    "position": {
      "type": "Point",
      "coordinates": [ 2.05, 57.1 ]
    }
  },
  "properties": {
    "status": "Idle"
  }
}
```

We use mainly the Thng Id in our Thing database (location informations are updated in SO Engine to limit external calls).

1.3.2 Redirection[5]

We use redirection to create a new tag linked to a thing : *[POST] https://tn.gg/redirections*

Example :

```
curl -H "Content-Type: application/json" \
  -H "Authorization: $OPERATOR_API_KEY" \
  -X                                                              POST
'https://api.evrythng.com/thngs/Uk6qwgSReXPat5awagfNpHsk/redirector'
\
  -d '{
    "defaultRedirectUrl": "https://google.com/{thngId}"
  }'
```

[5] https://developers.evrythng.com/v3.0/reference#section-create-a-thng-s-redirection

Which returns :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "createdAt": 1496668317836,
  "updatedAt": 1496668348257,
  "defaultRedirectUrl":
"https://google.com/Uk6qwgSReXPat5awagfNpHsk",
  "evrythngUrl":
"https://api.evrythng.com/thngs/Uk6qwgSReXPat5awagfNpHsk",
  "shortDomain": "tn.gg",
  "shortId": "799yeEi4qp",
  "hits": 0
}
```

So, we can read the url of the tag by *[GET] https://tn.gg/redirections?evrythngId=Id*

The tag url is then binded to the thing in the SO database.

1.3.3 Tag reading

For tag reading, we use the recognition API[6], with *POST /scan/identifications*

Tag reading is used in the SO mobile app, mainly for location change of things.

NOTE : we could use our own APIs for QR-code generation and reading, but this architecture is ready for the upcoming use of the TIS dynamic optical tags.

Conclusion

The main contribution of this project to SO Engine lies in the notion of Thing (or Thng), rather than product. This concept allows traceability of a specific object rather than a product category, allowing the full expressive power of SO Engine dynamic rules to be exploited.

We used here the QR Code tags (main TagItSmart object), but we will be able to extend without difficulty to other connected objects such as RFID tags (when APIs will be available on the TIS/EvryThng platform).

The following reports will describe in detail the SO Engine architecture, dynamic rules operation and usage scenarios.

[6] https://developers.evrythng.com/v3.0/reference#identifier-recognition