Overview of the Thing Description Directory

Victor Charpenay

March 16th, 2020

General Architecture

Requirements

The TD Directory (TDir) should be used for the **registration** and **discovery** of TD documents.

Specific requirements:

- Registration by Things of TD documents
- Registration by third-parties
- Deregistration and time-outs
- Simple discovery of Things (string matching or by interaction name)
- Advanced discovery (relationships between Things)
- Large-scale registration & discovery
- Access control

Design Choices

The TDir specification is the combination of two standards: the IETF Core Resource Directory[1] and the W3C Data Catalog (DCAT) vocabulary[2].

CoRE Resource Directory

Designed for constrained RESTful environments where the behavior of agents is driven by links and forms exposed by constrained devices, a Resource Directory (RD) "contains information about resources held on other servers, allowing lookups to be performed for those resources." The IETF specifies a standard Web interface for RDs.

DCAT Vocabulary

"DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web."

[1] https://tools.ietf.org/html/draft-ietf-core-resource-directory

[2] https://www.w3.org/TR/vocab-dcat/

Architecture Overview

The TDir specification provides a multi-protocol interface (CoAP, HTTP at least) to an RDF store, with some internal logic (e.g. for time-outs and access control).

Resource Directory Web Interface

Payload Formats

Payloads essentially encode links (with context, target and relation type). The default content type should be `application/link-format` but a TDir also supports RDF representations of links and forms (based on the TD model[3]).

```
<coap://light.local/state>;
ct=50;
rt="https://www.w3.org/2019/wot/td#readproperty"
```

Figure: CoRE Link format example

```
{
    "href": "coap://light.local/state"
    "op": "readproperty",
    "contentType": "application/json"
}
```

Figure: JSON TD example

[3] https://www.w3.org/TR/wot-thing-description/#sec-hypermedia-vocabulary-definition

Endpoints Overview

The registration and discovery of TD resources happens via a RESTful interface.

`/td`	POST	registration
`/td/{id}`	PUT, DELETE	TD document handle
`/td-lookup/ep`	GET	look-up by endoint
`/td-lookup/res`	GET	look-up by resource attributes
`/td-lookup/sem`	GET	semantic look-up (SPARQL)
`/td-lookup/frame`	GET	semantic look-up (JSON-LD frames)

Table: RD endpoints and allowed methods

Registration Endpoint

A TD document is first registered with a POST request to `/td`, to which the RDF responds with the URI of a newly created handle (built from the Thing's `id`). This handle can then be updated or deleted depending on the Thing's life cycle.

```
» POST /td {"id":"urn:ex:light", ...}
« 201 Created Location:/td/urn:ex:light
» PUT /td/urn:ex:light?lt=3600
« 204 No Content
» PUT /td/urn:ex:light {"id":"urn:ex:light","base":"http://192.168.1.5"}
« 204 No Content
» DELETE /td/urn:ex:light
« 204 No Content
```

Figure: Example of an exchange around registration between a Thing and a TDir

Discovery (Look-up) Endpoints

Simple discovery can be performed by searching for endpoints or resource attributes and more complex discovery can be performed with SPARQL queries and a prototypical GraphQL-like variant, using JSON-LD frames[4].

Examples of queries:

- by endpoint: what are the interaction affordances of `urn:ex:light`?
- by resource attributes: what interaction affordances expect JSON as input?
- using SPARQL: what are the interaction affordances that allow me to change the temperature of that room?
- using JSON-LD frames: *idem* (simpler syntax)

[4] `https://www.w3.org/TR/json-ld11-framing/`

RDF Dataset Description

DCAT Concepts

A dataset in DCAT is defined as a "collection of data, published or curated by a single agent, and available for access or download in one or more serializations or formats"

TD Model	DCAT Vocabulary
TD Document	Dataset
Interaction Affordance	(Simple) Resource
TD Directory	Catalog

Table: Correspondence between TD and DCAT concepts

DCAT Properties

Datasets (TD documents) can be annotated with metadata like: identifier, provenance, time of creation, time of last modification, access control policy.

```
<urn:ex:light> a dcat:Dataset ;
               dct:publisher <http://light.local> ;
               dct:identifier <urn:ex:light> ;
               dct:issued "2020-03-16T00:00:00Z" ;
               dct:modified "2020-03-16T00:00:00Z" ;
               odrl:hasPolicy :defaultPolicy .
```

Figure: Example of a TD document stored as a DCAT dataset

Use Case: Conference Room

TD Documents

WoT servients situated in a conference room register on a TDir the TD document they themselves host. See `https://www.vcharpenay.link/talks/urdf-wot.html`.

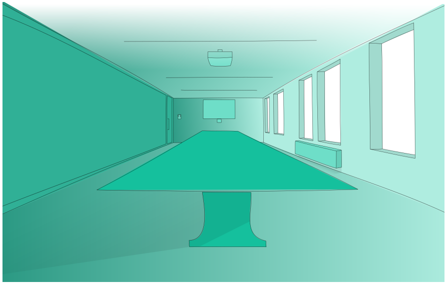

Figure: Example of a conference room with radiators, lights, electrical appliances, etc.

Discovery Sequence

Servients recognize each other by querying interaction affordances from Things in the same location and for the same physical property (temperature, illuminance, electrical energy).

```
SELECT ?href WHERE {
    <urn:ex:conference-room> bot:hasElement ?sensor .
    ?sensor a sosa:Sensor ;
            sosa:measures ?property ;
            td:hasInteractionOffordance ?affordance .
    ?property a saref:Temperature .
    ?affordance td:hasForm ?form .
    ?form hctl:hasTarget ?href .
}
```

Figure: SPARQL query example asking for all sensors measuring temperature in the conference room (see `http://prefix.cc` for prefix URIs)

Implementation

Thingweb Directory

A JVM implementation of the TDir specification is available on Github as `thingweb-directory`[5]

Implementation issues:

- no full implementation of JSON-LD 1.1 for the JVM
  - quick fix: store JSON TDs as RDF literal (but no proper SPARQL lookup)
- no CoAP interface yet
- no CoRE Link format (only JSON-LD/RDF)

[5] `https://github.com/thingweb/thingweb-directory`