HORIZONS 2020 PROGRAMME

Research and Innovation Action – FIRE Initiative

Call Identifier:	H2020–ICT–2014–1
Project Number:	643943
Project Acronym:	FIESTA-IoT
Project Title:	Federated Interoperable Semantic IoT/cloud Testbeds and Applications

Tools and Techniques for Managing Interoperable Data sets V1

Document Id:	FIESTA-IoT-D45-170315-Draft.doc
File Name:	FIESTA-IoT-D45-170315-Draft.pdf
Document reference:	Deliverable 4.5
Version:	Draft
Editor:	Aqeel Kazmi, Martin Serrano, Amelie Gyrard, Joao Bosco Jares
Organisation:	NUIG-DERI
Date:	15 / 03 / 2017
Document type:	Deliverable
Dissemination level:	PU

This document contains information, which is proprietary to the FIESTA-IoT Consortium.

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Martin Serrano	NUIG-DERI	2015/05/11	Initial Draft Proposal
	Amelie Gyrard Cassio Prazeres	NUIG-DERI	2015/07/16 2015/08/18	Move section from Deliverable brainstorming document D4.1 which are relevant for this task 4.3, Related work: Linked Open Graph tool, LDoW-PaN model (Cassio's section)
	Amelie Gyrard	NUIG-DERI	2015/09/15 2015/10/08 2015/10/19 2015/10/21	Improve Picture Description of the testbed repository Section Register Testbed (picture + demo), Section Testbed Discovery (picture + demo) Section Adding a new device to the testbed + section Storing the testbed RDF dataset in a database or triple store Storage testbed registered Java Data Objects (JDO) in a google database + future work section + refactoring section + testbed discovery section
	Amelie Gyrard	NUIG-DERI	2015/11/20 2016/03/11 2016/04/25	WebVOWL study with M3-lite, Tools to visualize resources, Related work user interfaces table of content
V02	Amelie Gyrard	NUIG-DERI	2016/06/27 2016/07/05	Integrate testbed and resource registration and discovery, Limitation of existing work
	Amelie Gyrard	NUIG-DERI	2016/09/16 2016/09/21	Update table of content with experimenter visualization, etc. BASIL tool for SPARQL templates
V03	Amelie Gyrard Joao Bosco Jares	NUIG-DERI	2016/11/04	New section Tutorial to help experimenters deal with the FIESTA-IoT endpoint and datasets, Generation of SPARQL query with Node-RED
V04	Tiago Teixeira Amelie Gyrard Tarek Elsaleh	Unparallel NUIG-DERI UNIS	2016/11/07 2016/11/09 2016/11/09	Section Validation Reference section + various improvements Section tutorial with yasgui SPARQL queries examples
	Paul Grace Rachit Agarwal Jorge Lanza	ITINNOV Inria UC	2016/11/11 2016/11/11 2016/11/15	Quality Review Technical review Technical review
V05 V06	Amelie Gyrard	NUIG-DERI	2016/11/16 2016/11/17 2016/11/22	Address comments provided by reviewers

V07	Amelie Gyrard	NUIG	2016/11/22	Address comments provided by reviewers
V08	Tiago Teizeira	Unparallel	2016/11/09	Addressing reviewers comments for Validation section.
V09	Aqeel Kazmi	NUIG	2017/02/19	Updates in Section 3 and Section 5, Addressing reviewers' comments, merging various versions + inputs provided by UNICAN
	Paul Grace Rachit Agarwal Jorge Lanza	ITINNOV Inria UC	….	Quality Review Technical review Technical review
V10	Aqeel Kazmi	NUIG	2017/03/09	Addressing Reviewer comments, preparing for submission
V11	Aqeel Kazmi	NUIG	2017/03/15	Access changes, generate version for submission
V12	Martin Serrano	NUIG	2017/03/15	Circulated for Approval
Draft	Martin Serrano	NUIG	2017/03/15	EC Submitted

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

Acronym	Meaning
API	Application Programming Interface
CKAN	Comprehensive Knowledge Archive Network
CRUD	Create, Read, Update and Delete
FOAF	Friend of a Friend
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things (IoT)
JDO	Java Data Objects
JSON	JavaScript Object Notation
LOD	Linked Open Data
LOG	Linked Open Graph
LOV	Linked Open Vocabularies
LOV4IoT	Linked Open Vocabularies for Internet of Things
M2M	Machine to Machine
M3	Machine-to-Machine Measurement
OWL	Ontology Web Language
QUDT	Quantities, Units, Dimensions and Data Types
RDF	Resource Description Framework
RDFS	RDF Schema
SenML	Sensor Markup Language
SSN	Semantic Sensor Networks
VoID	Vocabulary of Interlinked Datasets
W3C	World Wide Web Consortium
WoT	Web of Things
XML	Extensible Markup Language

1 EXECUTIVE SUMMARY

In FIESTA-IoT, we federate the Internet of Things (IoT) data stemming from a set of heterogeneous systems and their entity resources (such as smart devices, sensors, actuators, etc.). This vision of integrating IoT platforms, test-beds and their associated silo applications brings several scientific challenges; first there is a need to aggregate and ensure the interoperability of data streams stemming from different IoT platforms or test-beds, secondly, the need to provide tools and techniques for building applications that horizontally integrate diverse IoT Solutions, and above all, the need to specify and implement tools and techniques for testbed agnostic access to data sets stemming from multiple heterogeneous IoT platforms.

This deliverable describes the tools and techniques for accessing datasets via the FIESTA-IoT meta-cloud infrastructure in a *Testbed Agnostic* manner, i.e., this is independent and transparent to the underlying testbeds. These tools enable experimenters to get access to FIESTA-IoT compliant datasets through Graphical User Interface (GUI) or Application Programming Interfaces (APIs). This deliverable is based upon FIESTA-IoT Ontology (a semantic model designed in WP3 more explanations of which can be found in Deliverable 3.1.1 [9] and Deliverable 3.1.2 [10]). In order to provide additional functionalities required for the experimentation, mechanisms that reuse the FIESTA-IoT ontology to easily get access to datasets registered within the FIESTA-IoT platform have been implemented (e.g., access to metadata of the data streams / datasets). To this end, the interfaces (discussed in section 3 and 5) offer the means for: querying, accessing and processing data sources/streams at the level of the FIESTA-IoT meta-cloud; exploiting query languages and technologies (e.g., SPARQL) that are suitable for IoT/cloud testbeds (developed in WP3).

The structure of this deliverable is as follows, we first investigate background and related work regarding interoperable datasets in section 2. Then, we explain the resource registration carried out by testbeds providers and the resource discovery mechanisms utilised by experimenters to gain access to datasets in section 3. Then the validation process is explained to ensure the interoperability of datasets in section **Error! Reference source not found.**. Subsequently, testbed agnostic mechanisms are discussed together with examples in section 5 to guide users on how to interact better with the FIESTA-IoT. Finally, conclusions and future work are described in section 6.

1.1 Audience

This deliverable is addressed to the following audiences:

- **FIESTA-IoT researchers and developers:** notably individuals engaging in the development of the FIESTA-IoT platform. This deliverable will provide details which will help them to understand the design and usage of APIs to access datasets.
- **IoT application developers and solution providers** emphasizing on smart city applications using the IoT paradigm. Application developers and solution providers are expected to be interested into the interfaces discussed in this deliverable for accessing data streams.
- **IoT researchers:** notably researchers working on abstract service models for heterogeneous IoT applications and platforms. To these researchers, this deliverable may serve as a source of information to be useful for their research.
- **Experimenters and Open Call participants** can use this document to understand the technical details about the FIESTA-IoT tools that are designed for accessing heterogeneous datasets in testbed-agnostic manner.

2 BACKGROUND & RELATED WORK

In this section, we investigate background and related work concerning open datasets and the tools to utilise them. For this reason, we deeply study the existing work to see how to reuse, integrate or extend these tools and techniques according to the FIESTA-IoT requirements (see deliverable D2.1 “Stakeholders Requirements” [11]). We take inspiration from existing research and development carried out by the community to design and implement tools to satisfy the requirements of the FIESTA-IoT project.

2.1 Linked Open Data Cloud

At the beginning of the FIESTA-IoT project, the Linked Open Data (LOD) cloud[1] was considered as an inspiration so that it could be extended to IoT. LOD is a huge catalogue of datasets available on the web that we could reuse (after extending it to IoT) to build smarter applications. However, currently faced challenges with LOD include: finding, reusing, and combining existing datasets (a difficult task due to heterogeneous formats of the datasets). The concept of LOD cloud is similar to the task of testbed agnostic access datasets within the FIESTA-IoT platform.

In this section, we describe current research approaches exploiting the LOD data cloud which can guide us to build tools that can be used to get access to FIESTA-IoT datasets. Particularly we present and discuss:

[1] http://lod-cloud.net/

[2] http://datahub.io/

[3] http://watson.kmi.open.ac.uk/WatsonWUI/

- Linked data search engines as semantic browsers of datasets.
- Linked data visualization and some tools to assist users in finding datasets fitting their needs.

2.1.1 Linked Data Search Engines

Datahub[2], is a free and powerful data management platform from the Open Knowledge Foundation, based on the CKAN (Comprehensive Knowledge Network data management system). Datahub is a portal to get, use and share datasets represented using any format (CSV, RDF, etc).

Figure 1 shows datasets found when looking for "linked Sensor data". A dataset can be selected to get more description, have access to the SPARQL endpoint, dump file, etc. (see Figure 2). We took inspiration from such semantically annotated sensor datasets to design unification of sensor/IoT datasets provided by heterogeneous testbeds, resources and observations registered within the FIESTA-IoT platform. Based on the deep analysis of previous IoT projects dealing with IoT datasets, we realized the need of a common dictionary that we called taxonomy. This leads to the design of the M3-lite taxonomy, explained in Deliverable D3.1.2, which unifies sensor metadata descriptions by naming them in a similar way [10].

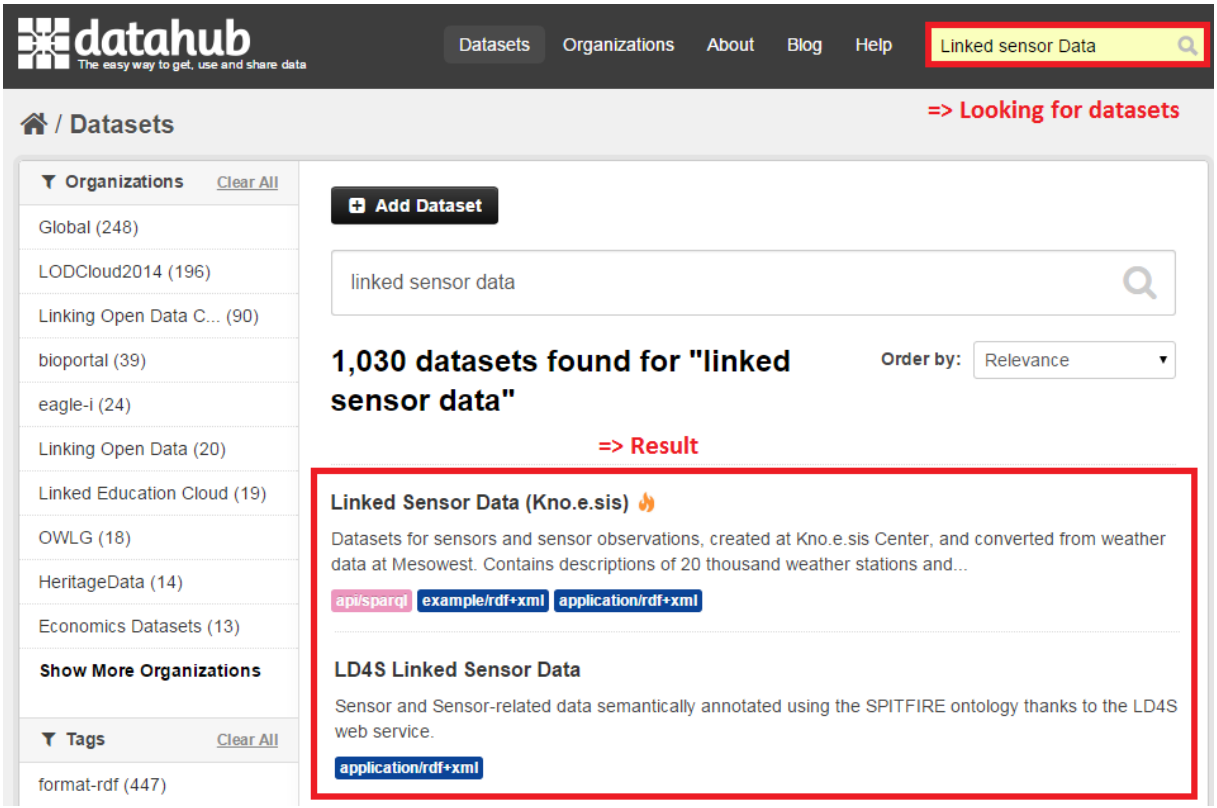


Figure 1: Datasets found on Datahub when looking for "Linked Sensor Data" datasets

[2] http://datahub.io/

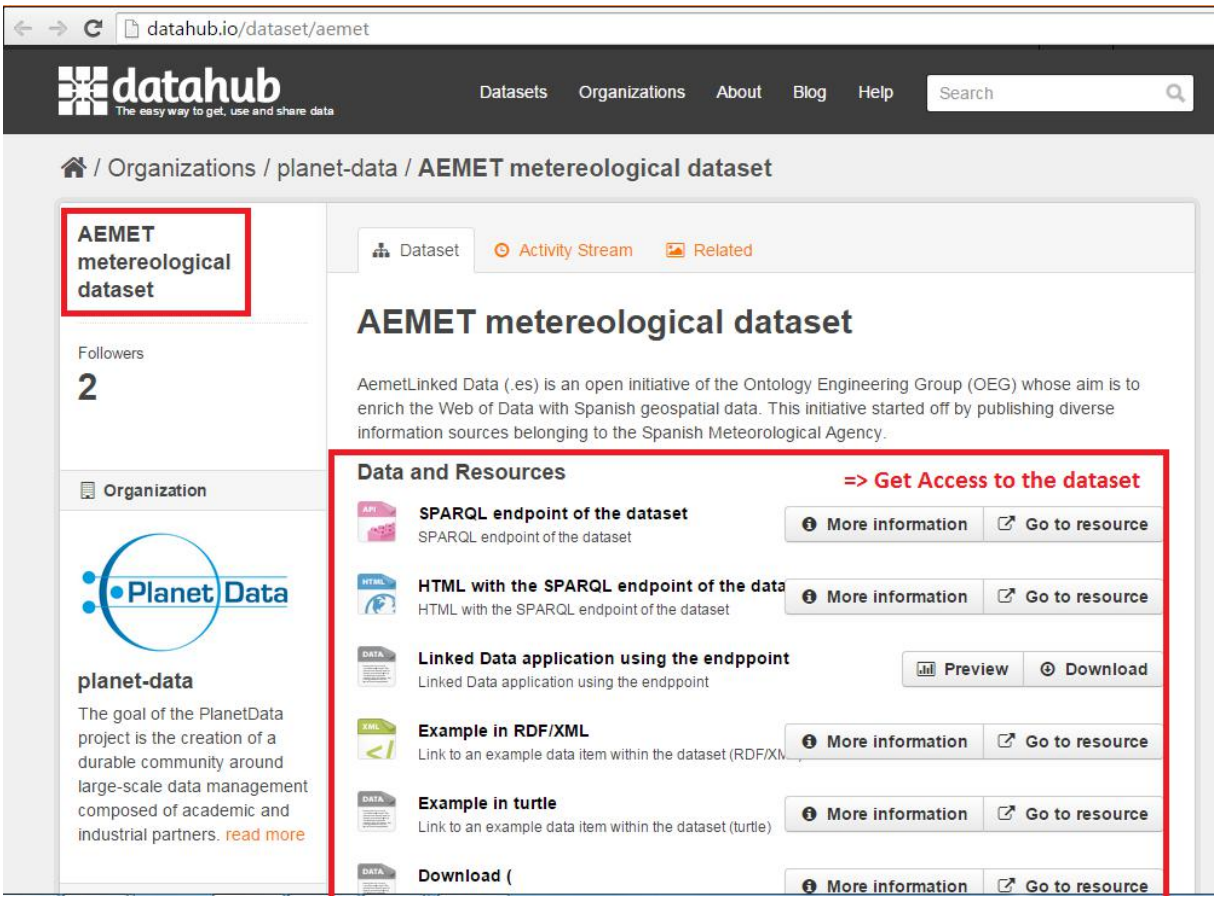


Figure 2: Description of a dataset (AEMET Meteorological dataset) on datahub

Watson[3] is a search engine exploring for ontologies and semantic documents [7]. Figure 3 shows the user interface to interact with Watson that also provides links to interact through APIs, or to submit URIs to be scanned. For instance, we could submit FIESTA-IoT ontologies and datasets to Watson. As an example, Figure 4 shows the result of Watson when searching for 'sensor' related ontologies and documents. The search result lists concepts (classes and their instances) that contain 'sensor' keyword either in the the URI or in the description.

Figure 3: The Watson semantic search engine Interface

[3] http://watson.kmi.open.ac.uk/WatsonWUI/

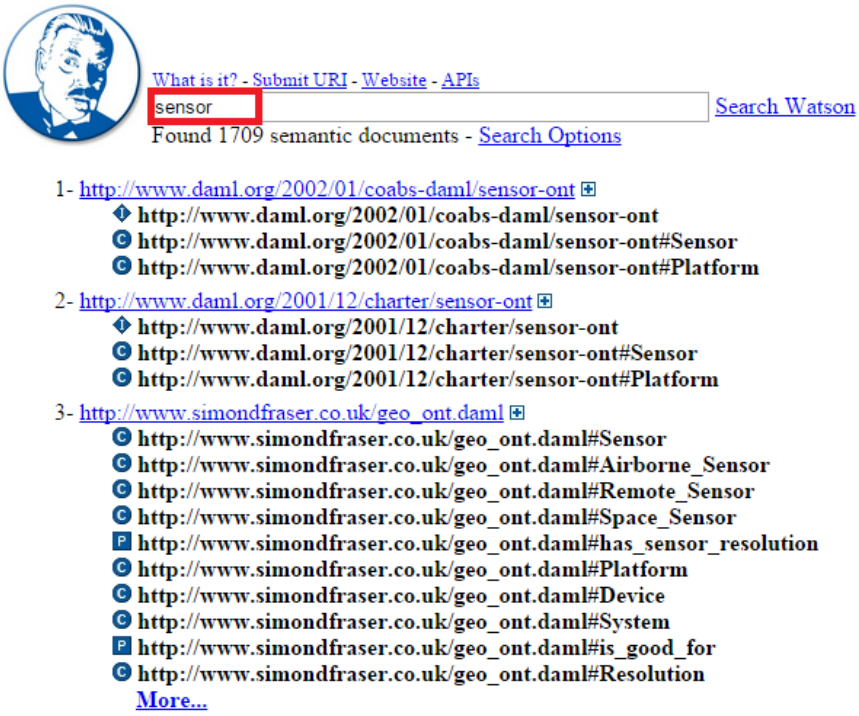


Figure 4: Result of the Watson search engine when the keyword was 'sensor'

Sindice is a Linked Data explorer [19] as depicted in Figure 5. It is a lookup index for Semantic Web documents, which can be used, for example, in IoT applications to find out relevant RDF sources. Now, it became a commercial product which demonstrates the usability and the scalability of the project. Taking inspiration from this tool, to explore IoT datasets registered within FIESTA-IoT is encouraged.

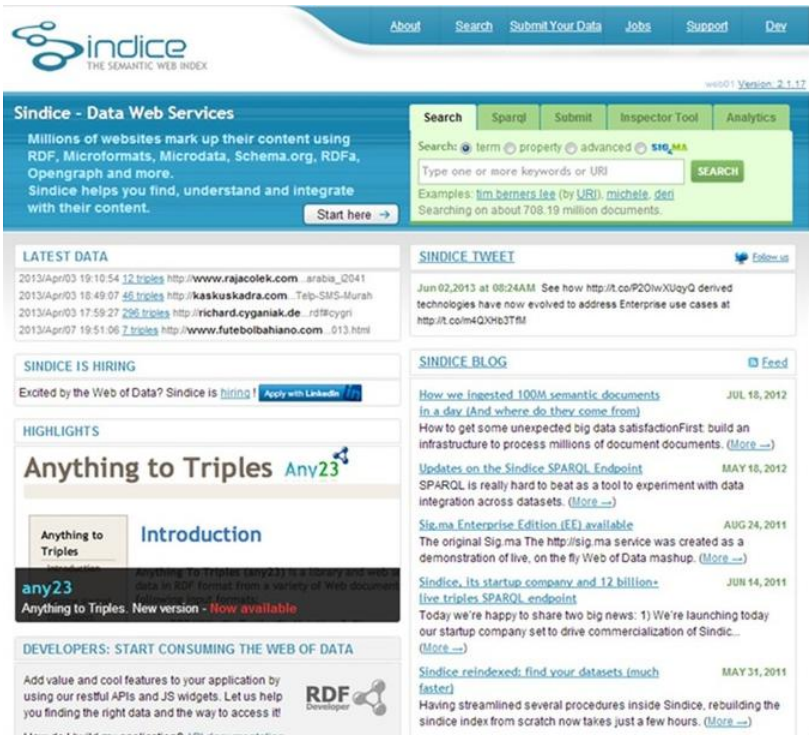


Figure 5: The Sindice link Data explorer

Swoogle[4] is another example of a semantic search engine. Swoogle can be used for searching ontologies and RDF documents [8] (see Figure 6). At the time of writing this deliverable, we did not find ontologies relevant for IoT domain. And It does not seem to be maintained anymore.

Figure 6: The SWOOGLE ontology search engine

2.1.2 Linked Data Visualization

Visualizing FIESTA-IoT datasets would improve the usability of the platform. For this reason, we explore existing linked data visualization tools that we may reuse and extend for IoT and FIESTA-IoT. There are existing efforts in the semantic web community working on linked Open Data Visualization [17]. Atemezing et al. design the Linked Data Visualization Wizard (LDVizWiz) to automatically generate visualizations [2] [3]. This work is focused on geospatial datasets. Figure 7 shows a tool to navigate through the Linked Open Data cloud[5]. When clicking on the bubble, we can access the dataset description as displayed in Figure 8. This is a user-friendly approach, something similar can also be made available within FIESTA-IoT platform. As discussed later, the design of WebVOWL tool is a first step towards enabling this methodology within the FIESTA-IoT.

[4] http://swoogle.umbc.edu/

[5] http://lod-cloud.net/versions/2014-08-30/lod-cloud.svg

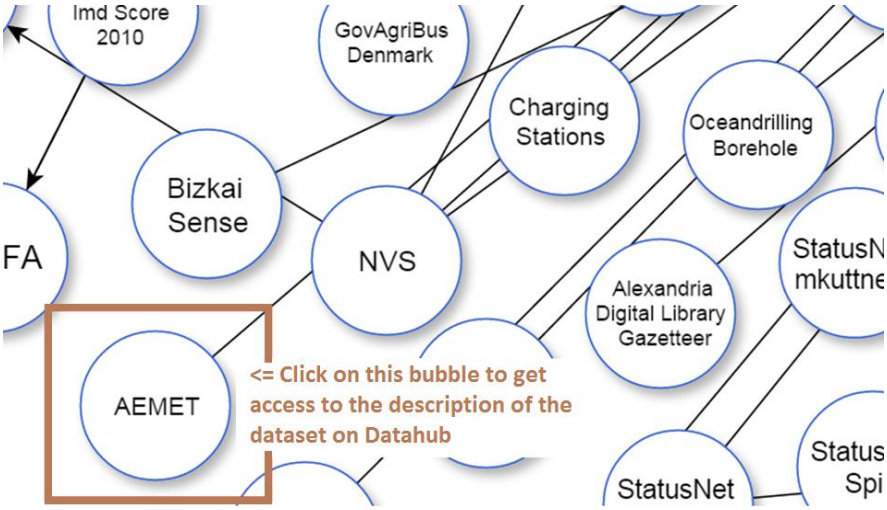


Figure 7: Navigation through the Linked Open data cloud

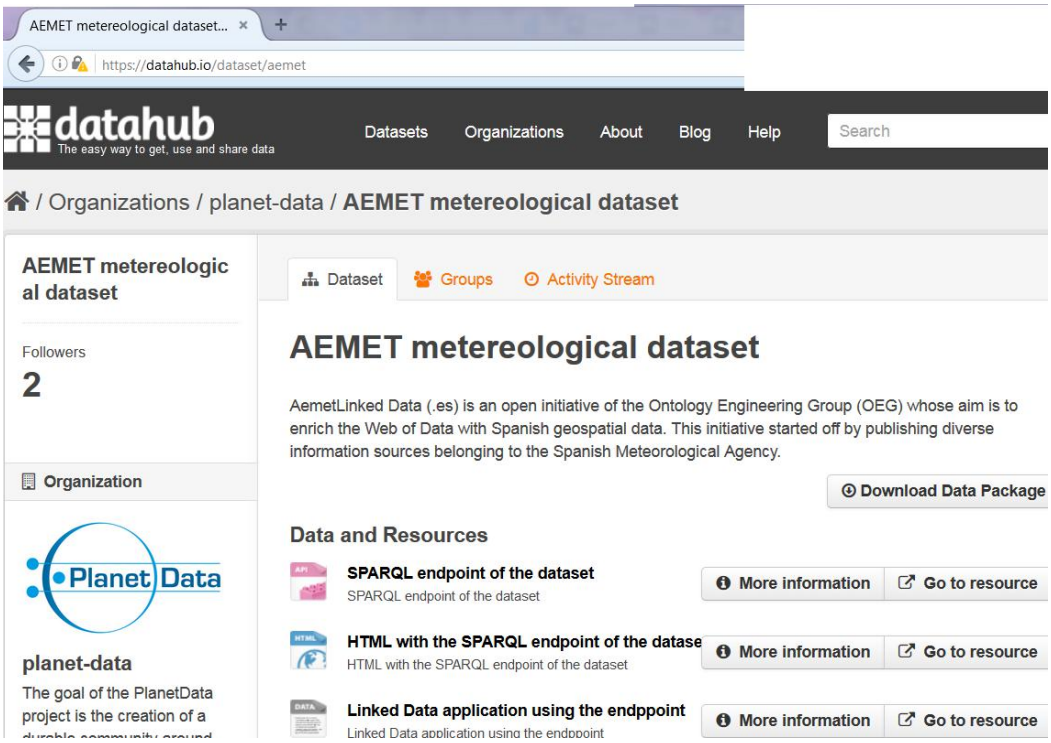


Figure 8: Get access to the AEMET dataset through the LOD interactive cloud

It would be useful to automatically generate a user-friendly interface according to the type of sensors. For instance, location data is usually represented on a map whereas temperature data in a chart. Logre et al. propose such automatic generation of sensor visualization according to the sensors employed [15] [16]. The work is based on the SenML format[6] used to describe sensor measurements.

Within the FIESTA-IoT project, current testbeds support for instance the JSON-LD format, but in the open calls, some testbeds might support the SenML format. For this reason, this work might be relevant to FIESTA-IoT as well.

[6] https://tools.ietf.org/html/draft-jennings-senml-10

2.1.3 Linked Open Graph (LOG)

Linked Open Graph (LOG)[7] is a visualization and exploration tool to browse and navigate aggregated data and SPARQL endpoints [5]. This tool has been used in the context of smart city to explore RDF smart city data annotated according to the KM4City ontology [4]. This tool has been tested and due to lack of functionalities provided as is aimed for a specific domain, we do not consider it to be a good fit within the FIESTA-IoT platform.

2.2 Ontology & Dataset Catalogue

2.2.1 Linked Open Vocabularies (LOV)

Linked Open Vocabularies (LOV)[8] is a dataset referencing more than 500 ontologies [20] (see Figure 9). We extended this catalogue for IoT by adding/inserting IoT related ontologies (added iot-lite, m3-lite, addition of fiesta-iot ontology is under process).

In LOV, a new ontology can be inserted only if they follow the LOV best practices:

- Ontology metadata to describe the ontology, the creator, the license, the create data, etc. [21].
- Labels and comments for each concept and property
- Referenceable URIs

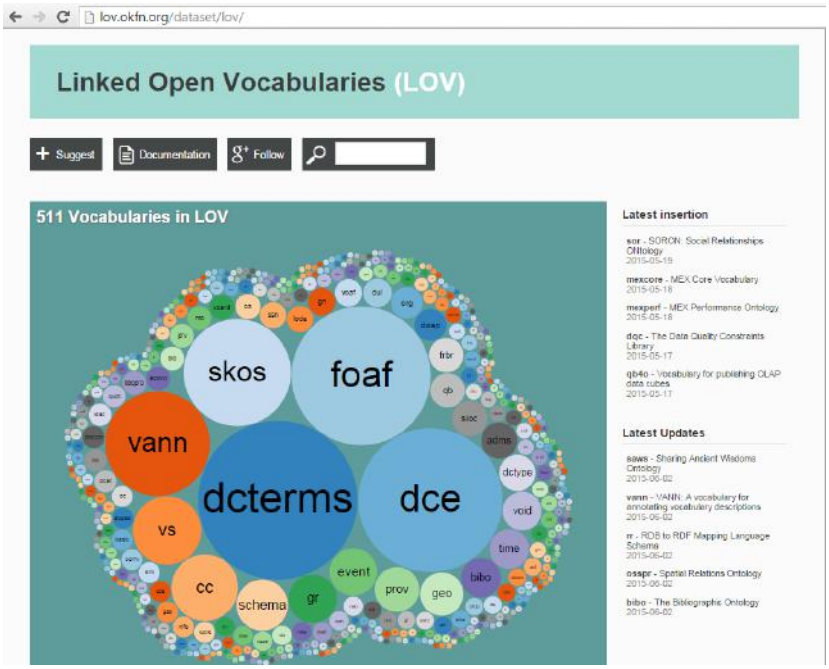


Figure 9: The Linked Open Vocabulary (LOV) catalogue

[7] http://log.disit.org/service/

[8] http://lov.okfn.org/dataset/lov/

2.2.2 Linked Open Vocabularies for Internet of Things (LOV4IoT)

The Linked Open Vocabularies for Internet of Things (LOV4IoT) is a dataset referencing more than 270 ontology-based projects relevant for IoT (see Figure 10, Figure 11) [13] and [14]. LOV4IoT classifies projects by domains. LOV4IoT referenced 19 domains: Smart Home, Smart Energy, Activity Recognition, Tourism, Transportation, Smart Agriculture, Weather Forecasting, Smart City, Sensor Networks, Internet of Things, Healthcare, Food/Restaurant, Affective Science Music, Environment, Fire Management, Security, Unit and Others. LOV4IoT can be accessed via HTML web page[9] as well as RDF dataset[10]. LOV4IoT is explained more in detail in Deliverable 3.1.1 [9]. LOV4IoT references each IoT-ontology based on which sensors are used, the URL for ontology, datasets, rules where available, technologies used.

Figure 10: The Linked Open Vocabularies for Internet of Things (LOV4IoT)

[9] http://sensormeasurement.appspot.com/?p=ontologies

[10] http://sensormeasurement.appspot.com/dataset/lov4iot-dataset

Smart city ontologies

Authors	Year	Paper	Url onto	Technologies	Sensors	Rules	LOV status
Park, Mail: 26/09/14, Response: 26/09/14	2014	Semantic reasoning with contextual ontologies on sensor cloud environment	Smart airport ontology, XOntology	Protege, SPARQL, Hadoop, HDFS, MapReduce		semantic reasoning with SPARQL	
Maria Poveda	2014	READY4SmartCities project	Ontologies and datasets for smart cities	<= dataset available on the web			
Lecue et al., STAR city Mail: 01/04/14	2013-2014	See papers below	Weather ontology Ontologies (weather, travel, etc.) and rules URL Congested/Free Road, Heavy/Stopped/Light Traffic Flow, Road Traffic Accident	W3C Time, W3C Geo, SWEET, DBpedia, not SSN, Jena TDB, CEL DL (Description Logic) reasoner, owl api 3.4.2, W3C Time onto, W3C Geo ontology, SWRL, Jena TDB as RDF store, HTML, CSS, Javascript Dojo toolkit D3, JQuery		Rules: Heavy traffic flow, light traffic flow (SWRL), SWRL OWL restrictions ontology	zip file, Mail: 01/04/14 for this issue, best practices in the same time
[STAR-CITY] Paper: Semantic traffic diagnosis with STAR-CITY: architecture and lessons learned from deployment in dublin, Bologona, Miami and Rio (ISWC 2014)							

Figure 11: Smart cities related ontologies referenced in LOV4IoT along with the URL, datasets using specific ontology, technologies used to create the ontology, etc.

2.2.3 Vocabulary of Interlinked Datasets (VoID) ontology

VoID is an ontology that is used to describe datasets [1]. VoID enables selections of appropriate datasets from a list of potential ones. VoID eases the task of the developers in the following way:

- Locating the dataset that contains relevant information (e.g., looking for data generated by a specific sensor).
- Finding out how the dataset can be programmatically accessed (e.g, SPARQL endpoint, RDF dump).
- Finding out the license associated with the dataset, making sure that the data is accessible under open-access license.
- Understanding the content of the dataset in order to perform an alignment with other datasets.

The VoID description explains:

- The content of the dataset (eg., which IoT domain or which sensors are associated to this dataset).
- The interlinking to other datasets.
- Vocabularies used in the dataset.

The main purpose of VoID is to encourage the reuse of ontologies and datasets and even the interlinking between datasets (an idea on which FIESTA-IoT is built). Furthermore, VoID is used by the Linked Open Vocabularies (LOV) catalogue and could be exploited by semantic search engines.

VoID editor[11] enables describing datasets and generating RDF code according to the VoID ontology (see Figure 12).

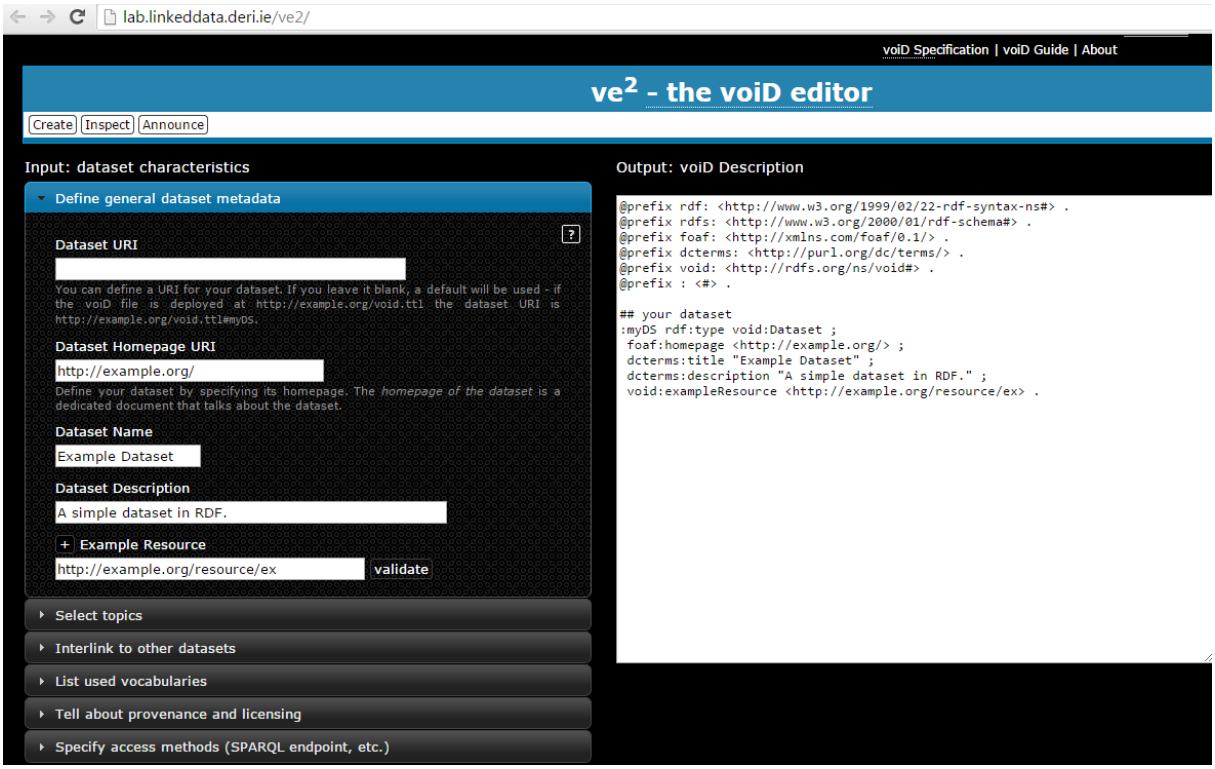


Figure 12: The VoID editor

Another inspiration from this work could be taken to extend the FIESTA-IoT ontology with the VoID ontology in order to interlink the IoT datasets in a better way. Furthermore, The semantic annotation could also be improved by using ontologies such as VoID to link datasets and OWL-S to describe services.

2.2.4 BASIL (Building APIs SImpLy)

The **BASIL (Building APIs SImpLy)**[12] framework combines REST principles and SPARQL endpoints in order to benefit from Web APIs and Linked Data approaches [6]. BASIL reduces the learning curve of data consumers since they query web services exploiting SPARQL endpoints. The main benefit is that data consumers do not need to learn the SPARQL language and related semantic web technologies.

2.3 User Interfaces

User interfaces are important in order to encourage the usability of the system. We investigate three UI tools namely: D3.js, WebVOWL, and LDoW-PaN that would enable nice visualization of FIESTA-IoT datasets.

[11] http://voideditor.cs.man.ac.uk/

[12] http://basil.kmi.open.ac.uk/app/

2.3.1 D3.js and D3SPARQL

D3.js[13] is a JavaScript library to visualize datasets in a user friendly interface. It proposes a large set of visualizations (see Figure 13). Another example library in this area would be the D3SPARQL, JavaScript library for visualization of SPARQL results [23]. D3SPARQL simply uses the D3.js library to visualize the results of a SPARQL query. The library provides a number of types to visualize the query results, such as, charts, graphs, trees, maps, tables, etc. D3.js and D3SPARQL are relevant for FIESTA-IoT for the following reasons:

- The bubble views seem relevant to display IoT datasets, as already done by the successful Linked Open Data community.
- The LOV catalogue uses also the bubble view provided by D3.js
- Zoom-able views seems to be fitting the FIESTA-IoT requirements.

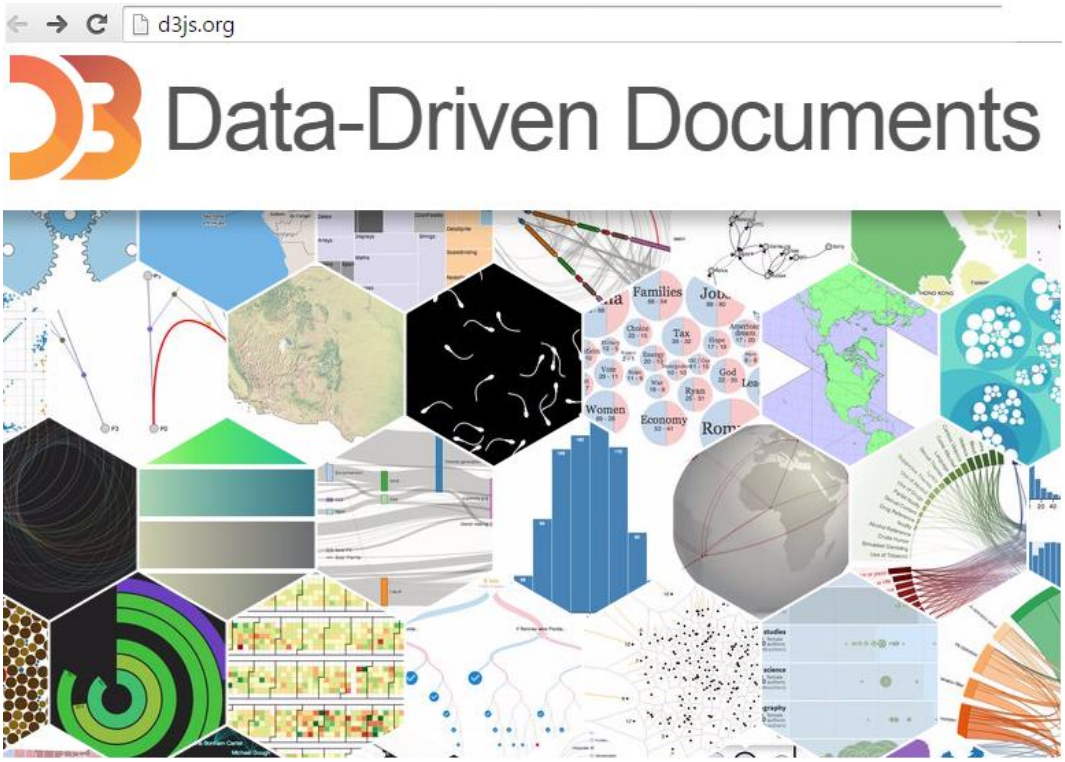


Figure 13: A large set of visualization is possible with D3.js

[13] http://d3js.org/

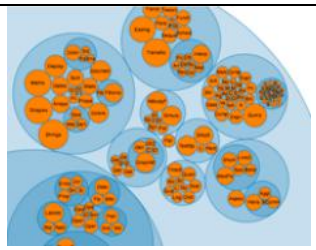
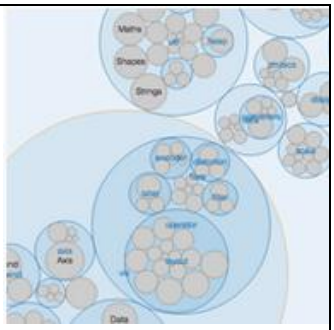
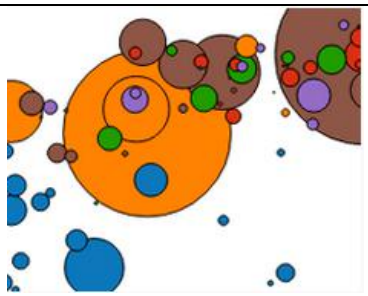
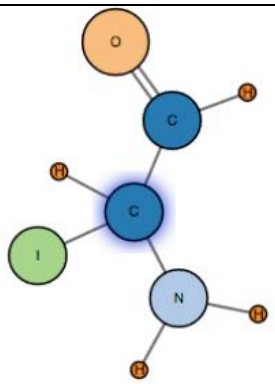
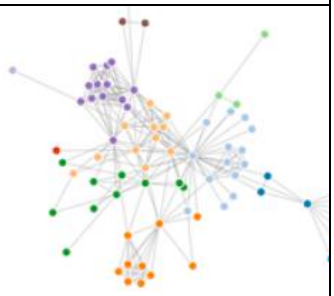
		
Bubble chart	Circle Packing	Zoomable pack layout
		
Motion Chart	OrgoShmorgo	Force-Directed Graph

Figure 14: Views relevant for the FIESTA-IoT project

2.3.2 WebVOWL

WebVOWL is used by Linked Open Vocabulary (LOV) and enables the online display of the schema of the ontology (see Figure 15 or visit online[14]). We also present a proof-of-concept illustration (see appendix II) to showcase the WebVOWL tool being used to visualize the FIESTA-IoT ontology including SSN, IoT-Lite and M3-lite taxonomy. This tool is relevant for ontology designer (e.g., the FIESTA-IoT consortium) since it enables fast ontology documentation in case the ontology itself is well structured and well documented. This tool is also relevant for experimenters, since it enables a first easy interaction with FIESTA-IoT ontology, which is frequently much more appreciated compared to reading documentation.

[14] http://vowl.visualdataweb.org/webvowl/#iri=http://spi-fm.uca.es/spdef/models/genericTools/itm/1.0

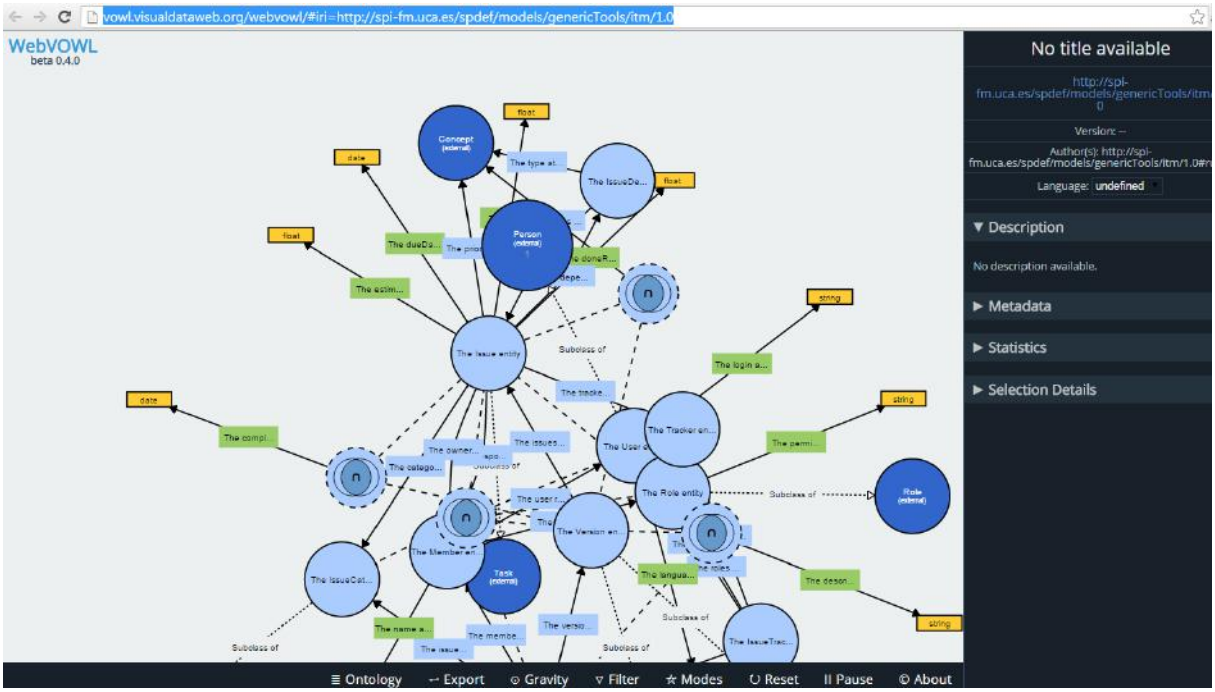


Figure 15: WebVOWL displaying ontologies online for documentation and easy navigation

2.3.3 LDoW-PaN

LDoW-PaN[15] is a Linked Data presentation and navigation model, based on the Dexter Model[16], focused on the average user. The presentation and navigation layer uses structures provided by the interface preparation layer to effectively present and provide navigation through data contained in the Dexter network produced during the process. The presentation and navigation layer is the highest-level layer of the LDoW-PaN model, and it implements the interface between the user and the network. The interface preparation and presentation and navigation layers are extensible. We can create new structures, new presentation and navigation ways. It can be adjusted to create more robust and adequate interfaces according to a system's needs (IoT Data, for example).

Actually, the LDoW-PaN model is implemented as following: i) a Web Service, which implements the four lower-level layers of the LDoW-PaN model; ii) a client-side script library, which implements the presentation and navigation layer; and iii) a browser extension, which uses these tools to provide Linked Data presentation and navigation for users browsing the Web. More description regarding LDoW-PaN is available in appendix I.

[15] https://github.com/WiserUFBA/LDoW-PaN/wiki

[16] https://github.com/dexter/dexter

2.4 Limitation and potential improvements of existing work

Indeed there are limitations in the literature that we studied. We have also taken some inspirations from existing works to satisfy the requirements to enable testbed agnostic access functionality in FIESTA-IoT platform:

- Visualization of datasets for easy navigation is important and increases usability of the platform, for example, Linked Data Visualization Wizard (LDVizWiz) is a good tool to visualize datasets. In FIESTA-IoT, we will provide a tool to help users to visualize and navigate though FIESTA-IoT datasets.
- Adapt/extend the VoID ontology and editor to generate RDF descriptions for IoT datasets or real stream.
- The LOV4IOT dataset should be extended with more IoT-related datasets. Moreover, there is a need to improve the LOV4IoT tool to add filter by domains, choose only the datasets that are already shared, ease the navigation.
- Based on our expertise, we realized that there is a real need to spread semantic web best practices within the IoT community in order to encourage the reusing of IoT ontologies already designed.
- Offer metadata and observations accessible through Web APIs.
- Enable query endpoints for developers to query datasets.
- Develop easy to use GUIs to access resources related testbeds.

We took inspiration from such semantically annotated sensor datasets available on the Linked Open Data Cloud to be able to design the unification of sensors provided by heterogeneous testbed, resources and observations registered within the FIESTA-IoT platform.

Based on our expertise, we realized that there is a real need to disseminate semantic web best practices to engage semantic interoperability within the IoT community in order to encourage the reusing of IoT ontologies already designed. For instance, numerous IoT ontologies cannot be loaded within tools such as WebVOWL, which hinder automation.

3 DISCOVERING INTEROPERABLE DATASETS

In the previous section, we listed and discussed a number of existing approaches harnessed to describe datasets and enable access to these datasets. A number of limitations are faced in existing works that are also highlighted. In this section, we explain how the users (such as experimenters) can discover available resources provided by the FIESTA-IoT platform. The registry and discovery mechanisms use semantic web technologies, more precisely the FIESTA-IoT ontology, for further details see [10] and [22].

We now discuss the IoT-registry component, developed specifically for FIESTA-IoT platform, which is used to describe the resources of registered testbeds. In addition, the measurements generated by the registered resources are also exposed by this component.

3.1 IoT Registry

The FIESTA-IoT meta-directory, the component that is in charge of the semantic resource descriptions from the federated testbeds, as well as the observations that these resources produce is exposed through the, so called, IoT-Registry. It basically wraps the triple-store that internally holds the aforementioned FIESTA-IoT meta-repository. The IoT-Registry has been implemented as a REST web-service, the IoT-Registry provides a set of *Create, Read, Update and Delete (CRUD)* interfaces that enables access to the meat-repository supported by a Jena-based[17] triple-store.

This component exposes the single-entry point where all the testbeds must register their IoT Resources' descriptions (i.e. the information used to describe the underlying devices deployed in the testbeds) and all observations produced by these devices are kept. Moreover, all FIESTA-IoT components access the semantic datasets through it, also.

In order to keep the testbed agnostic nature that the FIESTA-IoT meta-directory has to show, the annotated descriptions, after being syntactically and semantically validated, are "anonymized". This process of "anonymizing" basically consists in overwriting the bindings that points to the original testbeds' domains included in the original annotated documents containing all the resource and observation descriptions. These new identifiers, independently of which testbed they originally belong to, are the ones exposed through the federation graph. In addition to this, any entity from the resulting federation linked-data graph can be directly accessed through their referenceable FIESTA-IoT identifier, thus accessing to its semantic description. The core concept behind the IoT-Registry is to exploit the intrinsic linked-data nature of semantics. This way we are offering a real *Web of Things* approach.

Besides the above-depicted functionality, the IoT-Registry includes a query interface that supports the execution of SPARQL queries. This offers a flexible and standard interface to access the information stored in the triple-store (i.e. federated testbeds' resource descriptions and observations).

Below, the main REST resources are summarized. They are, afterwards, described more thoroughly in Table 1, Table 2, Table 3 and Table 4. As this document focuses

[17] https://jena.apache.org/

on the techniques for testbed agnostic access to data-sets, the tables only include the procedures for accessing information available in the meta-repository. The API supports both read and write operations, as shown in the API documentation[18]; however, for the sake of experimenters consuming data, the part of the API that is of interest is the one described below.

- `/resources`: Endpoint through which all the semantically annotated information linked to the testbed Resources (i.e. description, phenomenon measured, unit of measurement, location, etc. of the devices registered by each of the federated testbeds) are provided.
- `/observations`: Endpoint through which all the semantically annotated information linked to the observations or measurements gathered by testbed Resources (i.e. value and unit of the measurement, phenomenon measured, location, timestamp, etc.) are provided.
- `/testbeds`: Endpoint that enables quick access to semantic descriptions belonging to a particular testbed.
- /`queries`: Endpoint that enables the execution of SPARQL queries on a specific graph (observation or resources) or the whole triple-store. It provides the possibility to store SPARQL templates for later use.

The implementation of the IoT-Registry has tried to follow the approach whenever possible, but in order to provide a more user-friendly interface, some of the functionalities are not fulfilling this philosophy. The REST web-service implemented is accepting and returning RDF documents in various formats, like JSON-LD, N3, RDF/XML, CSV, plain text, n-quads, etc.

Before presenting all the services supported by the IoT-registry API, it is worth highlighting that many of them are restricted for internal (i.e. administration) usage.

Table 1: IoT-registry Resources-related services

`/resources?original_id=boolean&class=string` `original_id`: Flag to set whether to provide original or testbed agnostic identifier `class`: Class IRI of the returned entities (filtering)		
Method	Description	Request Body
GET	List all available entities in the resources triple-store	
POST	Register a resource (or array of resources) into the IoT-registry triplestore	Semantically annotated document compliant with FIESTA-IoT ontology
`/resources/{id}`		
Method	Description	Request Body
GET	Retrieve a specific resource description as an RDF document	

[18] http://platform.fiesta-iot.eu/iot-registry/docs

DELETE	Delete a stored semantic entity associated to resources	
/resources/{id}/original_id `original_id`: Flag to set whether to include original or testbed agnostic identifier		
Method	Description	Request Body
GET	Retrieve the resource original identifier assigned by the testbed it belongs to	

Table 2: IoT-registry Observations-related services

/observations?original_id=boolean&class=string `original_id`: Flag to set whether to provide original or testbed agnostic identifier `class`: Class IRI of the returned entities (filtering)		
Method	Description	Request Body
GET	List all available entities in the observations triple-store	
POST	Register an observation (or group of them) into the IoT-registry triplestore	Semantically annotated document compliant with FIESTA-IoT ontology
/observations/{id}		
Method	Description	Request Body
GET	Retrieve a specific observation as an RDF document	
DELETE	Delete a stored semantic entity associated to observations	
/observations/{id}/original_id `original_id`: Flag to set whether to include original or testbed agnostic identifier		
Method	Description	Request Body
GET	Retrieve the resource original identifier assigned by the testbed it belongs to.	

Table 3: Testbeds IoT Service Endpoint

/testbeds		
Method	Description	Request Body
GET	List all available testbeds	
/testbeds/{id}		

Method	Description	Request Body
GET	Retrieve the semantic testbed entity representation (`ssn:Deployment`)	
`/testbeds/{id}/resources?original_id=boolean` `original_id`: Flag to set whether to include original or testbed agnostic identifier		
Method	Description	Request Body
GET	Retrieve the resources registered by the testbed {id}.	

Table 4: IoT-registry Queries-related services

`/queries/store`		
Method	Description	Request Body
GET	List all the queries stored	
POST	Create a new query	Query document, including SPARQL, name, description, etc.
`/queries/store/{id}`		
Method	Description	Request Body
GET	Retrieve an stored query	
PUT	Update a query	The new query document, including SPARQL, name, description, etc.
DELETE	Delete a stored query	
`/queries/execute/{resources/observations/global}`		
Method	Description	Request Body
POST	Execute a SPARQL query on a specific graph or triple-store. (`resources` for Resource descriptions; `observations` for measurements; `global` for the whole meta-repository)	The SPARQL query.
`/queries/execute/{resources/observations/global}/id`		
Method	Description	Request Body

GET	Execute a stored SPARQL query on a specific graph or triple-store. (`resources` for Resource descriptions; `observations` for measurements; `global` for the whole meta-repository)	

This section provided an overview and discussed the IoT Registry component used in the FIESTA-IoT platform.

3.2 Linked Open Data for Internet of Things Visualization

Here we discuss briefly how to display the registered IoT datasets available within the FIESTA-IoT platform in a user-friendly manner to ease the task of users to access and use IoT datasets. The implementation could be done using D3.js or even D3SPARQL.js in order to ease the usability of the FIESTA-IoT platform. We have chosen specific visualizations to display the datasets such as bubble view, table view, the way the testbeds are described, color preferences, etc. Instead of using a drop-down list, we could use a bubble view as depicted in Figure 16.

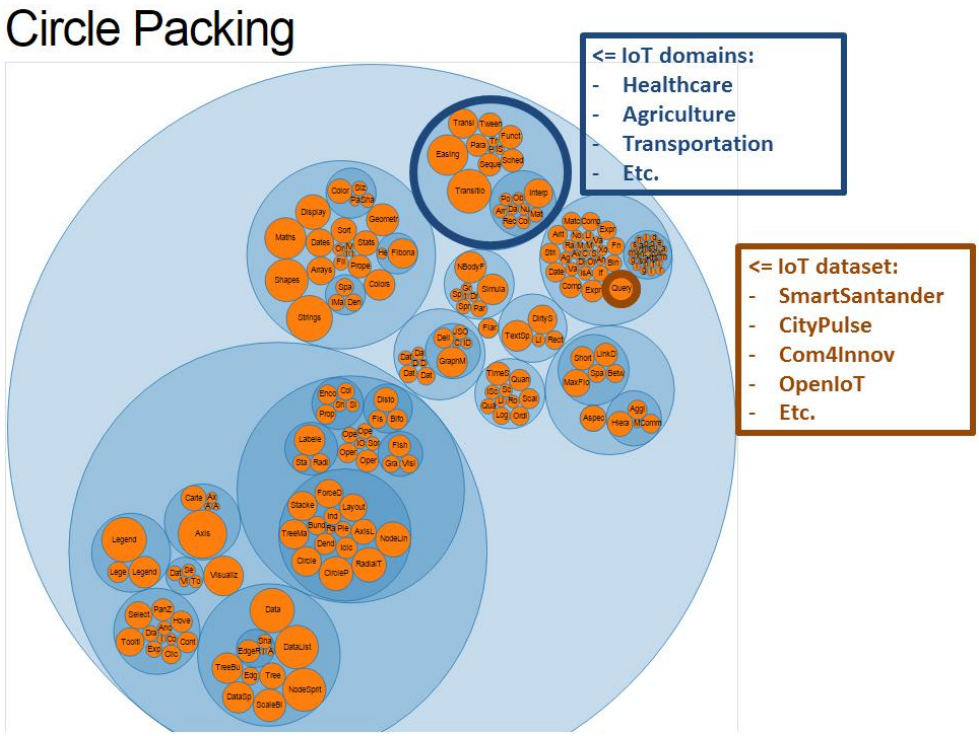


Figure 16: Visualization of IoT datasets (e.g., testbeds) classified by IoT applicative domains

4 TESTBED AGNOSTIC ACCESS MECHANISM VALIDATION PLAN - LAB TESTING

The FIESTA-IoT platform must be able to provide information in a way that is independent and transparent to the registered testbeds. A lab-testing validation plan is proposed that provides a controlled environment where, through several steps, it tests the FIESTA-IoT platform capabilities of accessing testbeds' information in an agnostic manner. This access mechanisms will be validated by doing verification on the produced response, with the purpose of validating its coherence with the request made.

The objective of the FIESTA-IoT platform is to combine data from multiple sources as a response of a single request. However, the FIESTA-IoT platform is only able to deal with data represented by the FIESTA-IoT ontology and since each testbed produces data with different formats, types, etc., the connection between the registered testbeds and the platform must be made through an annotator. This adapter is going to map the data available in the testbeds with the FIESTA-IoT ontology, therefore enabling the FIESTA-IoT platform to use it.

The dataset obtained through the FIESTA-IoT platform can be a combination of information obtained from the available (and registered) testbeds. In the proposed validation plan, we first populate FIESTA-IoT platform with "dummy" testbeds. We then compare the information present in the dataset, with the information returned by the execution of the selected query. After this comparison, it will be possible to know if the dataset contains exactly the information that it is supposed to contain.

This validation plan defines several steps to ensure that the FIESTA-IoT platform possesses agnostic access mechanisms to testbeds:

1. As this is a Lab testing, the first step is to install FIESTA-IoT on the environment dedicated for Lab testing, allowing to have an independent platform running, where we can execute a completely autonomous set of tests, and be able to reproduce specific results to adjust possible incoherencies.
2. The next step is to create several "dummy" testbeds (i.e. between 5 and 10). These "dummy" testbeds will basically be databases with a pre-defined dataset. The plan is to have each of the "dummy" testbeds to have a data format similar to the actual in-house testbeds, and have a small subset of their data stored.
3. The third step would be to create/use an annotator for each of the created "dummy" testbeds. The annotators will map each testbed's data into the FIESTA-IoT format. As each of the "dummy" testbeds will have a data format like the in-house testbeds, the idea is to use the already existing annotators (or part of it, as the testbeds will not be recreated entirely for this validation).
4. Step number 4 will be to register the "dummy" testbeds into the FIESTA-IoT platform using the already available registration tools. This will ensure that the FIESTA-IoT platform is aware of the existing "dummy" testbeds. With each new registered "dummy" testbed, more data is going to be available to the FIESTA-IoT platform, therefore increasing the available datasets for the execution of this validation process.

5. With the FIESTA-IoT platform local instantiation completely deployed and working, with the "dummy" testbeds registered with a pre-defined set of test data covering several sensors, parameters, locations, etc., there is the need to define several queries with the purpose of asking the FIESTA-IoT platform for information.
6. With the queries defined, there is the need to execute them in the local FIESTA-IoT Platform and retrieve the result. For example: "*Give me all the atmospheric temperature data available in Lisbon*" (mapped in an equivalent SPARQL query). Based on this request the FIESTA-IoT platform is going to retrieve information from all the "dummy" testbeds containing thermometers sensors located in Lisbon, and provide it as the result. At the same time, we will search the datasets of each of the "dummy" testbeds and identify which temperature data should be included in the result for this specific query.
7. The last step would be to compare both results, the one provided by FIESTA-IoT platform, with the data collected from the "dummy" testbeds (in FIESTA-IoT format), and the result of the manual search. This comparison will check if the result provided by FIESTA-IoT platform is exactly is expected, enabling us to validate the FIESTA-IoT testbed agnostic access to testbeds.

The purpose of this lab-testing validation is to ensure that the data accessed on the FIESTA-IoT platform is retrieved in an agnostic way, i.e. independently of the testbeds. Figure 17 shows an overview of how this validation will occur.

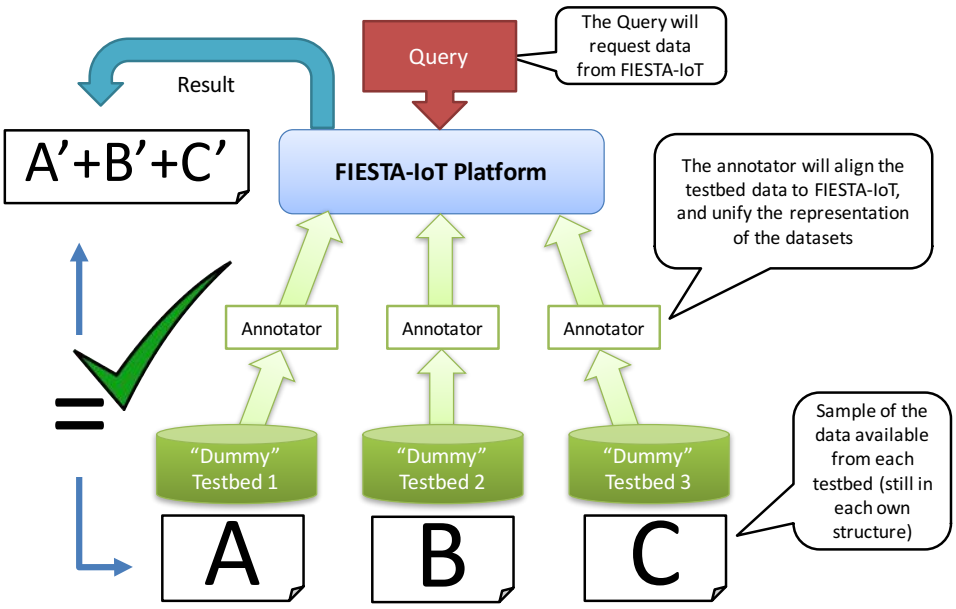


Figure 17: Overview of the lab-testing validation of the testbed agnostic access mechanism

When information is being requested from the FIESTA-IoT platform it must be ensured that the user will have access to the correct dataset regardless of the query made. During this validation, the process is going to be repeated several times with different configurations, queries, etc. to ensure that the FIESTA-IoT access mechanisms are testbed agnostic.

5 TESTBED AGNOSTIC ACCESS MECHANISM

In this section, we provide further detail on the testbed agnostic access mechanism developed for the FIESTA-IoT platform together with some examples to help the users and experimenters understand the usability of these tools and processes. First we discuss the semantic web access to registered datasets within the FIESTA-IoT platform. Then the SPARQL endpoint to FIESTA-IoT datasets is discussed and query examples along with their results are provided. At the end of the section, we introduce a novel technique that is based on the Node-RED environment for automatically generating the SPARQL queries.

5.1 Semantic web access to FIESTA-IoT datasets

As discussed in section 3, in order to provide testbed agnostic access to the information stored in the FIESTA-IoT Meta-Directory, the original identifiers or IRIs to the different semantic entities that the various testbeds register are hidden by the IoT-Registry. The new IRIs generated are always referenceable through the Web of Things created by the Meta-Directory and its IoT-Registry interface. Thus, the semantic Resource/Observation descriptions can be directly accessible through Internet in a full-blown web-linked-data manner. This way, any FIESTA-IoT exported element can be linked from any semantic document, enabling the paradigm of a fully semantic web.

For example, Text box 1 shows a GET request for all the resources whose class is `ssn:Device`. This also includes entities in the graph which are of any of that class' defined sub-classes (e.g. `ssn:SensingDevice`). The result of that request is a list of IRIs of all the FIESTA-IoT resources fulfilling this constraint.

Request

```
GET
http://platform.fiesta-iot.eu/iot-registry/resources?class=http%3A%2F%2Fpu
rl.oclc.org%2FNET%2Fssnx%2Fssn%23Device
Accept: application/json
```

Response

```
200 OK
Connection:  keep-alive
X-Powered-By:  Undertow/1
Server:  WildFly/10
Content-Type:  application/json
Content-Length:  827
Date:  Wed, 16 Nov 2016 12:27:50 GMT

{
  "resources":
  "http://platform.fiesta-iot.eu/iot-
registry/resources/NpgGzP9dG8xotY0plBozJTkuqD9dAcVcxr7pw8Vxf9-
```



```
l1tGbuuJKannjqlEfvyWFjBWt-
sZIv9xZE5yEaG7ZT8mbIpC9_Iz_TLHfa1qIQ7BQiwphmX4SEXfMsXWK-zsF",
  "http://platform.fiesta-iot.eu/iot-
registry/resources/YE793JIklp12Hx70tBxxG-
Rziujsl00wKjf4YtLVf7Lj7r0nrSPIlFHScHX_TeWNAgNn-
RXSdW7cUEkdRfzwp1XApa9XWtZdRbUJvjGI3bvZOpYzBIKk5POGLEma2kLtZYltOWbF2zQyYmB
V3eHygVCP5HEiDvO3-BdkWn_FZK0=",
  "http://platform.fiesta-iot.eu/iot-
registry/resources/oU4v6zlJfxkewzqZwNx0H3VzHq6_FImTq7UzexAG2l2M6DqKn2JVvxW
6SO5S7kfUM8EvDKt-KAQqk-
SvZxSdwaN7AIZq6yjLisrsHwZv0BfIUmwQAP_5Iw2nO6PbKaMMPIrd5pCl4eEf6CNBFsegYfTG
FPg_gYY5Qg_tVhPqWWQ=",
  "http://platform.fiesta-iot.eu/iot-
registry/resources/xygdi5A1trEHwSV_Tqbmro6Rbx0cE-
HbXTl0wT9LHp9Gq1YL78duzCWe9O_HRy1vCOEzR2htIOS0ahKGwI6PtThJEKgxVLiyHekHX2Tr
bxCRguFXoYpk7ttxetFluHmDReCQetAvJ8MBUsyHAPeQTA=="
  ]
}
```

Text box 1: HTTP GET Request listing all stored `ssn:Device` resources

Making these FIESTA-IoT IRIs referenceable, we can directly retrieve these resource description by simply executing an HTTP GET request addressed to the Resource IRI's URL. Text box 2 shows an illustrative example using JSON-LD as the RDF serialization format. In this case, the Resource is a `ssn:Device` that embraces various properties, i.e. `ssn:hasDeployment`, `ssn:hasSubSystem` and `ssn:onPlatform`.

Request

```
GET
http://platform.fiesta-iot.eu/iot-
registry/resources/AhSelezuW9KWV8RNwnZylX6Ii2G-
QSds0J7THkEbv9G3eI2rIfj1T5IstOqa9iFuQbp8fOb3zlGR7SV_zqHY1kFAQDSa9ZN22RZTSi
kzLR46GUzKGEyH98_kN0kOno8J
Accept: application/ld+json
```

Response

```
200 OK
Connection:  keep-alive
X-Powered-By:  Undertow/1
Server:  WildFly/10
Content-Type:  application/ld+json
Content-Length:  1644
Date:  Wed, 16 Nov 2016 12:24:42 GMT

{
  "@id": "http://platform.fiesta-iot.eu/iot-
registry/resources/AhSelezuW9KWV8RNwnZylX6Ii2G-
QSds0J7THkEbv9G3eI2rIfj1T5IstOqa9iFuQbp8fOb3zlGR7SV_zqHY1kFAQDSa9ZN22RZTSi
```

```
kzLR46GUzKGEyH98_kN0kOno8J",
  "@type": "http://purl.oclc.org/NET/ssnx/ssn#Device",
  "hasDeployment": "http://platform.fiesta-iot.eu/iot-
registry/resources/y2yVmoA3DcTUnokIp9KF8HaF3FDp0Zewjq150T5PV-
lHCtZzMhOiSwWcxnQzt1Wj9eAJNkssoQuWaXbfuGs3LA==",
  "hasSubSystem": [
    "http://platform.fiesta-iot.eu/iot-
registry/resources/d1ym4VqtNkxYV33gILwXM3X3mSY3Ftb8IT-dVcfMNCvSTUF1kIHeQ-
AACXnr_29pFnv7WTRvFMyvEUdtS_0JDLrYZRVX8xt2gNrwAW0dLGUjgVOOqMOJgHyL8-
0oJVACWCx8QMhTl_gYLydOp8iN3w==",
    "http://platform.fiesta-iot.eu/iot-registry/resources/8dJHjD-
Ud8g6OfTkEOr1RGD6AyEnyfOxzh65cRt_aF2c48HcOSM3vvEYinQ4R05ZU5zp3W8yPAHOGCtUZ
vWq5ZGtUMdo7R9r1B-gmD-yOQzvJCEt-PBiYcwqsxJGlObn"
  ],
  "onPlatform": "http://platform.fiesta-iot.eu/iot-
registry/resources/6GLfYH4y7TC4l9dbTxQBUv7Q7aKVRHMRh95y-
jLThCzP0Qe0gnBXrJXOggM2w5aTR44BST_7RkeF9S94SOyfK8_b5ejKS6OyMth98gKcSHVCLBS
aP3TaGaaNoB76c1ni",
  "@context": {
    "onPlatform": {
      "@id": "http://purl.oclc.org/NET/ssnx/ssn#onPlatform",
      "@type": "@id"
    },
    "hasSubSystem": {
      "@id": "http://purl.oclc.org/NET/ssnx/ssn#hasSubSystem",
      "@type": "@id"
    },
    "hasDeployment": {
      "@id": "http://purl.oclc.org/NET/ssnx/ssn#hasDeployment",
      "@type": "@id"
    }
  }
}
```


Text box 2: HTTP GET Request retrieving a resource semantic description

Each of the IRIs in this Resource description are pointing to the semantic description of the Resource it is identifying. This way, FIESTA-IoT is providing a very simple way of browsing its triple store by using pure web technologies.

5.2 SPARQL query interface to FIESTA-IoT datasets

SPARQL is known to be the most common and widely used RDF query language. Therefore, it is sensible to offer a fully-fledged SPARQL interface, as part of the IoT-registry module that allows this kind of semantic queries on the FIESTA-IoT platform. In this sense, as presented in Text box 3, we have not only implemented a direct SPARQL endpoint, but also a system for the storage and execution of off-the-shelf SPARQL query templates. This additional functionality would make it easier to share knowledge between experimenters or testbeds and smooth the learning curve when it comes to cope with the FIESTA-IoT ontology and its underlying dataset representation. Next, the operation of these two main mechanisms is depicted.

The `/queries/execute` endpoint is a conformant SPARQL protocol service as defined in the SPARQL Protocol for RDF (SPROT). It allows users to query a

knowledge base via the SPARQL language. Results are returned in any of the common data representation formats, namely JSON, XML, CSV, etc.

The default endpoint runs the query on the "global" FIESTA-IoT semantic database, including both Resources and the linked Observations. However, it is also possible to limit the scope of the query to just the Resources or the Observations graph by identifying the graph in the request URL.

Text box 3 shows an example request to the `/queries` endpoint and its response. The request body, in a plain text format, contains the raw SPARQL sentence to be executed. In case the query is not compliant with the SPARQL format, i.e. there is a syntax error, the server will report the corresponding error. Otherwise, the successful execution returns a list of rows, each one with as many columns as variables were requested, fulfilling the query executed (in this particular example, only the `ssn:Device`).

Request

```
POST http://platform.fiesta-iot.eu/iot-registry/queries/execute/global
Accept: application/json
Content-Type: text/plain

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>

SELECT ?entity
WHERE {
  ?entity rdf:type ?type.
  ?type rdfs:subClassOf* ssn:Device.
}
```

Response

```
200 OK
Connection:  keep-alive
X-Powered-By:  Undertow/1
Server:  WildFly/10
Content-Type:  application/json
Content-Length:  954
Date:  Wed, 16 Nov 2016 13:02:37 GMT

{
  "vars": [
    "entity"
  ],
  "items": [
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/AhSelezuW9KWV8RNwnZylX6Ii2G-
QSds0J7THkEbv9G3eI2rIfj1T5IstOqa9iFuQbp8fOb3zlGR7SV_zqHY1kFAQDSa9ZN22RZTSi
```

```
kzLR46GUzKGEyH98_kN0kOno8J"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/xV9C0b3LuKXY9XDQow31E8mlYw_Y8uggFnUrYVRe2uGiCW5tzbv7jHM
pR3xQ3l7VLf9CwG-IvcFLFwUlsVpDpjvOwjnwJkMQan0L0rpfUCOjdRQLwWNr3-Yv_3NXq8qX"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/eTvUD260TiVD2ldEZSmo9sUKx8grud5gSreeeAUdUZtRaEwhWuCZbG0
vatj5MWffjVWRjswGD76bFVU5Otf-WrLeSA_WtQyQmokHqBV5OJQDV4iWsbf4CIMH9Y12etiQ"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/4Mvv0aMFqFmKSUby8gQWE4pGAUfa7BHnv-
gVTSzvg3sy9LAo0fjLPuha7bZUG7WNu1ggsfHN14Ynqn3-
ES06k4mY09fQ6KMH1lZWSBhNKcxehlHf_lNlLfgx6t10myfn"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/xmYW5-xX3TC5mE5umn-
UuvrT0ExlbDigimvzywa24pwl7jfxKVhyD5GFOqCOAuaX"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/8dJHjD-
Ud8g6OfTkEOr1RGD6AyEnyfOxzh65cRt_aF2c48HcOSM3vvEYinQ4R05ZU5zp3W8yPAHOGCtUZ
vWq5ZGtUMdo7R9r1B-gmD-yOQzvJCEt-PBiYcwqsxJGlObn"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/d1ym4VqtNkxYV33gILwXM3X3mSY3Ftb8IT-dVcfMNCvSTUF1kIHeQ-
AACXnr_29pFnv7WTRvFMyvEUdtS_0JDLrYZRVX8xt2gNrwAW0dLGUjgVOOqMOJgHyL8-
0oJVACWCx8QMhTl_gYLydOp8iN3w=="
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/6lImpfh2RzQVigSUMA6r73swm1sdDmb6DKemAFjFJmnirp2z7_wNZJd
5Sfy9C8801ZiZxzZA1Bof0MiOgsBa1APSFH0snxv39rAKB65aAI2Le-BBAZmgPRfiU_ZFNjKQ"
    }
]
```


Text box 3: HTTP POST Request retrieving stored ssn:Device using a SPARQL query

Aside this direct execution of SPARQL queries, we offer the possibility of storing SPARQL queries for a latter execution. And not only for personal purposes, but for sharing these, thus giving rise to a sort of “crowd-sourced” catalogue. Moreover, this option reduces the overhead and eases the action of executing multiple times the same SPARQL sentence.

Each of these query objects have the format shown in Text box 4. As can be seen, it includes a friendly name, a short description, the envisaged scope for the query (it

can be overridden) and the actual SPARQL query. Following the CRUD approach, they can be updated and deleted.

```
{
  "name":"ssn:Device subclasses retrieval",
  "description": "SPARQL query for retrieving all entities belonging to
ssn:Device class and its subsclasses",
  "value": "PREFIX rdf: <http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-
ns#>\r\nPREFIX rdfs: <http:\/\/www.w3.org\/2000\/01\/rdf-
schema#>\r\nPREFIX ssn:
<http:\/\/purl.oclc.org\/NET\/ssnx\/ssn#>\r\n\r\nSELECT ?entity
?type\r\nWHERE {\r\n  ?entity rdf:type ?type.\r\n  ?type rdfs:subClassOf*
ssn:Device.}",
  "scope" : "RESOURCES"
}
```

Text box 4: Example FIESTA-IoT query object

Besides, we have also extended the functionality including a proprietary template format that provides the possibility of adapting/changing the stored SPARQL on each request, thus leading to flexible on-demand queries. Users can specify parameters directly in the SPARQL query. To achieve so, some variables will be replaced with input parameters in the GET/POST requests based on a set of conventions. The syntax is based on a wildcard (i.e. %%%) which delimits a parameter name (and will never appear in a regular SPARQL query). Adding the parameter as a URL query parameter will replace the assigned value before executing the query. The requests included in Text box 5 demonstrates the functionality just described. Initially, we register a query object including a FIESTA-IoT SPARQL template, which aims at retrieving entities stored from any `ssn` class (see parameter `%%%class%%%`). The next step is launching a HTTP GET to the `/execute` endpoint indicating the `id` of the stored query object and assigning the desired value to the `class` parameter (in this case `Device`). As can be seen, we obtain a similar result to the one retrieved through the direct SPARQL (Text box 3).

1) Request

```
POST http://platform.fiesta-iot.eu/iot-registry/queries/store
Accept: application/json
Content-Type: application/json


{
  "name": "ssn generic subclasses retrieval",
  "description": "SPARQL query for retrieving all entities belonging to a
class and its subclasses",
  "scope": "GLOBAL",
  "value": "\nPREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#>\nPREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\nSELECT
DISTINCT ?entity ?type\nWHERE {\n  ?entity rdf:type ?type.\n  ?type
rdfs:subClassOf* <http://purl.oclc.org/NET/ssnx/ssn#%%%class%%%>\n}\n"
}
```

1) Response


```
200 OK
Connection:  keep-alive
X-Powered-By:  Undertow/1
Server:  WildFly/10
Content-Type:  application/json
Content-Length:  443
Date:  Wed, 16 Nov 2016 13:29:03 GMT

{
  "id": 15,
  "value":   "\nPREFIX   rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-
ns#>\nPREFIX     rdfs:     <http://www.w3.org/2000/01/rdf-schema#>\nSELECT
DISTINCT  ?entity  ?type\nWHERE  {\n   ?entity  rdf:type  ?type.\n   ?type
rdfs:subClassOf* <http://purl.oclc.org/NET/ssnx/ssn#%%%class%%%>\n}\n",
  "name": "ssn generic ssn subclasses retrieval",
  "description": "SPARQL query for retrieving all entities belonging to a
ssn class and its subclasses",
  "scope": "GLOBAL",
  "created": "2016-11-16T13:29:02+0200",
  "modified": "2016-11-16T13:29:02+0200",
  "vars": [
    "class"
  ]
}
```

2) Request

```
GET http://platform.fiesta-iot.eu/iot-registry/queries/execute/15?class=Device
Accept: application/json
```

2) Response

```
200 OK
Connection:  keep-alive
X-Powered-By:  Undertow/1
Server:  WildFly/10
Content-Type:  application/json
Content-Length:  954
Date:  Wed, 16 Nov 2016 13:02:37 GMT

{
  "vars": [
    "entity"
  ],
  "items": [
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/AhSelezuW9KWV8RNwnZylX6Ii2G-
QSds0J7THkEbv9G3eI2rIfj1T5IstOqa9iFuQbp8fOb3zlGR7SV_zqHY1kFAQDSa9ZN22RZTSi
kzLR46GUzKGEyH98_kN0kOno8J"
```

```
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/xV9C0b3LuKXY9XDQow31E8mlYw_Y8uggFnUrYVRe2uGiCW5tzbv7jHM
pR3xQ3l7VLf9CwG-IvcFLFwUlsVpDpjvOwjnwJkMQan0L0rpfUCOjdRQLwWNr3-Yv_3NXq8qX"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/eTvUD260TiVD2ldEZSmo9sUKx8grud5gSreeeAUdUZtRaEwhWuCZbG0
vatj5MWffjVWRjswGD76bFVU5Otf-WrLeSA_WtQyQmokHqBV5OJQDV4iWsbf4CIMH9Y12etiQ"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/4Mvv0aMFqFmKSUby8gQWE4pGAUfa7BHnv-
gVTSzvg3sy9LAo0fjLPuha7bZUG7WNu1ggsfHN14Ynqn3-
ES06k4mY09fQ6KMH1lZWSBhNKcxehlHf_lNlLfgx6t10myfn"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/xmYW5-xX3TC5mE5umn-
UuvrT0ExlbDigimvzywa24pwl7jfxKVhyD5GFOqCOAuaX"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/8dJHjD-
Ud8g6OfTkEOr1RGD6AyEnyfOxzh65cRt_aF2c48HcOSM3vvEYinQ4R05ZU5zp3W8yPAHOGCtUZ
vWq5ZGtUMdo7R9r1B-gmD-yOQzvJCEt-PBiYcwqsxJGlObn"
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/d1ym4VqtNkxYV33gILwXM3X3mSY3Ftb8IT-dVcfMNCvSTUF1kIHeQ-
AACXnr_29pFnv7WTRvFMyvEUdtS_0JDLrYZRVX8xt2gNrwAW0dLGUjgVOOqMOJgHyL8-
0oJVACWCx8QMhTl_gYLydOp8iN3w=="
    },
    {
      "entity": "http://platform.fiesta-iot.eu/iot-
registry/resources/6lImpfh2RzQVigSUMA6r73swm1sdDmb6DKemAFjFJmnirp2z7_wNZJd
5Sfy9C8801ZiZxzZA1Bof0MiOgsBa1APSFH0snxv39rAKB65aAI2Le-BBAZmgPRfiU_ZFNjKQ"
    }
]
```


Text box 5: Example FIESTA-IoT executing a stored template

Last, but not least, in addition to the API, we foresee to develop a portal for visualization and easy interaction with this ecosystem. This work will be reported in the second version of this deliverable.

5.3 Generating SPARQL queries with Node-RED

Within FIESTA-IoT platform, experimenters can also use Node-RED to automatically build the SPARQL queries that are compliant with the FIESTA-IoT ontology. This is illustrated in Figure 18. Text areas such as “Quantity Kind”, “List of Domains”, “List of Units” are related to the terms that have been introduced within the FIESTA-IoT

ontology. Other text areas such as value, location and longitude enable to modify generic parameter by concrete values within the generic SPARQL query.

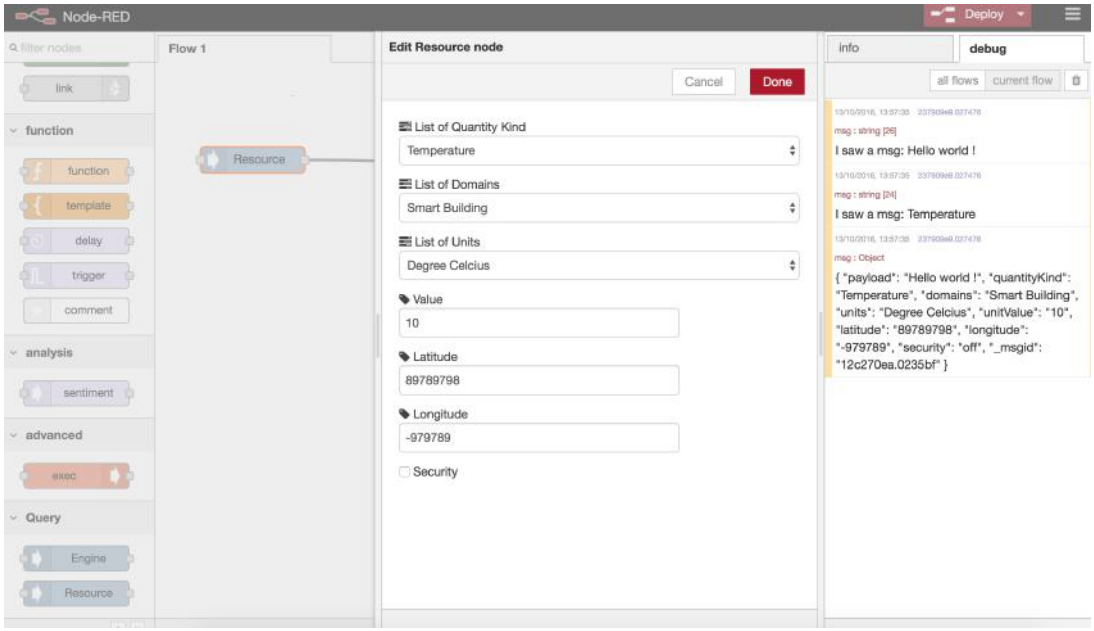


Figure 18. Node-RED tool used to generate the SPARQL query compliant with the FIESTA-IoT ontology

Generic SPARQL queries are used whose parameters are replaced with actual values when an experimenter creates the flows. For example, the generic SPARQL query in Text box 6 enables to query all "Resources" measuring a particular phenomenon:

```
PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
PREFIX m3-lite: <http://purl.org/iot/vocab/m3-lite#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
SELECT *
WHERE {
  ?resource a ssn:SensingDevice.
  ?resource iot-lite:hasQuantityKind ?quantityKind
}order by asc(UCASE(str(?s)))
```

Text box 6: Example SPARQL generic query

Parameters are automatically replaced by Node-RED. In this example, we replace **?**quantityKind parameter. By selecting the Temperature QuantityKind within the Node-RED, the SPARQL query replaced the **?**quantityKind parameter by the corresponding value selected. The drop-down list within Node-RED display the subclasses of QuantityKind referenced within the M3-lite taxonomy. By replacing the generic parameter **?**quantityKind, Node Red generates a Specific SPARQL query compliant with the FIESTA-IoT ontology to ease the task of experimenters (picture below). The following query (Text box 7) requests all Resources measuring a particular phenomenon (e.g. Temperature):

```
PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
PREFIX m3-lite: <http://purl.org/iot/vocab/m3-lite#>
```



```
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
SELECT *
WHERE {
  ?resource a ssn:SensingDevice.
  ?resource iot-lite:hasQuantityKind <http://purl.org/iot/vocab/m3-lite#Temperature>
}order by asc(UCASE(str(?s)))
```

Text box 7: Example query requesting all temperature resources

The same mechanism can be applied to other text areas provided by Node-RED. The following (Text box 8) generic SPARQL query deals with latitude and longitude:

```
PREFIX geo:  <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>

SELECT ?dev ?resource ?lat ?long
WHERE {
   ?dev a ssn:Device .
   ?dev ssn:hasSubSystem ?resource.
   ?resource a ssn:SensingDevice.
   ?dev ssn:onPlatform ?platform .
   ?platform geo:location ?point .
   ?point geo:lat ?lat .
   ?point geo:long ?long .
   FILTER (
     (xsd:double(?lat) >= xsd:double(?latMinFromNodeRed) )
   && (xsd:double(?lat) <= xsd:double(?latMaxFromNodeRed) )
   && ( xsd:double(?long) < xsd:double(?lontMinFromNodeRed) )
   && ( xsd:double(?long) > xsd:double(?lontMaxFromNodeRed) )
   )
}
```

Text box 8: Example query requesting all latitude and longitude resources

The query below (Text box 9) shows how to “Query all Resources within a geographical area”, the parameters ?latMinFromNodeRed, ?latMaxFromNodeRed, ?lontMinFromNodeRed and ?lontMaxFromNodeRed are being replaced by the values given on Node-RED.

```
PREFIX geo:  <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>

SELECT ?dev ?resource ?lat ?long
WHERE {
   ?dev a ssn:Device .
   ?dev ssn:hasSubSystem ?resource.
   ?resource a ssn:SensingDevice.
   ?dev ssn:onPlatform ?platform .
   ?platform geo:location ?point .
```



```
    ?point geo:lat ?lat .
    ?point geo:long ?long .
    FILTER (
      (xsd:double(?lat) >= "0"^^xsd:double)
    && (xsd:double(?lat) <= "60"^^xsd:double)
    && ( xsd:double(?long) < "100"^^xsd:double)
    && ( xsd:double(?long) > "-0.60"^^xsd:double)
    )
}
```

Text box 9: Example query consisting of multiple parameters.

In this section, we introduced Node-RED used for another purpose. From the experimenter/user's perspective, a specific node is dedicated to automatically generate the SPARQL query based on the user interface designed within the Node-RED itself. In this deliverable, we provided an overview of the SPARQL query generation work, this mechanism is actively being developed. Further work and the progress made in this regard will be reported in the second and final version of this deliverable.

6 CONCLUSION AND FUTURE WORK

We investigated the state of the art to deal with datasets such as the Linked Open Data Cloud and related approach such as Linked Open Vocabularies designed by the semantic web community to help us to design a common approach within FIESTA-IoT that is specifically harnessed for the Internet of Things domain. We also investigated tools that can be used to easily visualize datasets.

In this deliverable, we discussed the specifications and implementation of tools and techniques for testbed agnostic access to data sets stemming from multiple heterogeneous IoT platforms. We reuse some tools such as Node-RED, WebVOWL, D3SPARQL, some of them designed by the semantic web community, apply them and extend them within FIESTA-IoT platform to assist experimenters in dealing with the FIESTA-IoT datasets. Tools and techniques have been investigated and designed to get access to FIESTA-IoT datasets. For instance, Node-RED has also been extended to automatically generate SPARQL queries compliant with the FIESTA-IoT ontology. As a reminder, within the FIESTA-IoT platform, Node-RED is extended for three different use cases: 1) discoverability of resources provided by testbeds (see Deliverable D3.3.2), (2) generating an experiment description compliant with FedSpec Node Red (see Deliverable D4.1.1), and (3) generating a SAPRQL query with Node-RED. Then, a validation process has been presented within this deliverable. Another validation process regarding the semantic annotation of data compliant with the FIESTA-IoT ontology is explained in Deliverable 6.1 [FIESTA-IoT D6.1 2016]. Finally, appendices include some of the proof-of-concept work that has been carried out related to mechanism for testbed access agnostic to datasets within the FIESTA-IoT platform.

As a future work, we have in mind to improve the integration of the tools within the FIESTA-IoT core platform. The tools presented in above sections or explained in this deliverable and the set of web services will be actively improved and integrated in the platform. These advancements in the design and the work with regard to implementation will be reported in the second and final version of this deliverable which is due in the month 30 of the project.

7 REFERENCES

[1] Keith Alexander and Michael Hausenblas. Describing linked datasets-on the design and usage of void, the vocabulary of interlinked datasets. In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09, 2009.

[2] Ghislain Auguste Atemezing and Raphael Troncy. Towards interoperable visualization applications over linked data. In Talk Given at the 2nd European Data Forum (EDF), Dublin, Ireland (April 2013), http://goo. gl/JhVrax.

[3] Ghislain Auguste Atemezing and Raphaël Troncy. Towards a linked-data based visualization wizard. In Workshop on Consuming Linked Data, 2014.

[4] Pierfrancesco Bellini, Monica Benigni, Riccardo Billero, Paolo Nesi, and Nadia Rauch. Km4city ontology building vs data harvesting and cleaning for smart-city services. Journal of Visual Languages & Computing, 25(6):827–839, 2014.

[5] Pierfrancesco Bellini, Paolo Nesi, and Alessandro Venturi. Linked open graph: browsing multiple sparql entry points to build your own lod views. Journal of Visual Languages & Computing, 25(6):703–716, 2014.

[6] Enrico Daga, Luca Panziera, and Carlos Pedrinaci. A basilar approach for building web apis on top of sparql endpoints. 2015.

[7] Mathieu d'Aquin and Enrico Motta. Watson, more than a semantic web search engine. Semantic Web, 2(1):55–63, 2011. http://watson.kmi.open.ac.uk/WatsonWUI/.

[8] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In Proceedings of the thirteenth ACM international conference on Information and knowledge management, ACM, 2004.

[9] FIESTA-IoT, Deliverable D3.1.1. "Semantic models for testbeds, interoperability and mobility support and best practices" May 2016

[10] FIESTA-IoT, Deliverable D3.1.2. "Semantic models for testbeds, interoperability and mobility support and best practices" November 2016

[11] FIESTA-IoT, Deliverable D2.1. " Stakeholders Requirements" July 2015

[12] FIESTA-IoT, Deliverable D3.2.1. "Specification and implementation of common Testbed interfaces" July 2016

[13] Amelie Gyrard, Christian Bonnet, Karima Boudaoud and Martin Serrano. "LOV4IoT: A second life for ontology-based domain knowledge to build Semantic Web of Things applications". 4rd International Conference on Future Internet of Things and Cloud (FiCloud 2016), Vienna, Austria

[14] Amelie Gyrard, Ghislain Atemezing, Christian Bonnet, Karima Boudaoud and Martin Serrano. "Reusing and Unifying Background Knowledge for Internet of Things with LOV4IoT". 4rd International Conference on Future Internet of Things and Cloud (FiCloud 2016), 22-24 August 2016, Vienna, Austria.

[15] Ivan Logre, Sébastien Mosser, Philippe Collet, and Michel Riveill. Sensor data visualisation: a composition-based approach to support domain variability. pages 1–16, 2014.

[16] Sébastien Mosser, Ivan Logre, Nicolas Ferry, and Philippe Collet. From Sensors to Visualization Dashboards: Need for Language Composition. In Globalization of Modelling Languages workshop (GeMOC'13), Miami, 29 September 2013, pages 1–6. IEEE, 2013.

[17] Oscar Peña, Unai Aguilera, and Diego López-de Ipiña. Linked open data visualization revisited: A survey. Semantic Web Journal.

[18] Rohloff, Kurt and Dean, Mike and Emmons, Ian and Ryder, Dorene and Sumner, John. "An evaluation of triple-store technologies for large data stores". OTM Confederated International Conferences" On the Move to Meaningful Internet Systems. 2017

[19] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. Springer, 2007.

[20] Pierre-Yves Vandenbussche, Ghislain A Atemezing, Mará Poveda-Villalón, and Bernard Vatant. Lov: a gateway to reusable semantic vocabularies on the web. Semantic Web Journal, 2015.

[21] Pierre-Yves Vandenbussche and Bernard Vatant. Metadata recommendations for linked open data vocabularies. Version, 1:2011–12, 2011.

[22] Rachit Agarwal, David Gomez Fernandez, Tarek Elsaleh, Amelie Gyrard, Jorge Lanza, et al.. Unified IoT Ontology to Enable Interoperability and Federation of Testbeds. *3rd IEEE World Forum on Internet of Things*, Dec 2016, Reston, United States.

[23] Toshiaki Katayama. D3SPARQL: JavaScript library for visualization of SPARQL results. Proceedings of the 7th International Workshop on Semantic Web Applications and Tools for Life Sciences, 2014

APPENDIX I – LDOW-PAN

This section was planned to be included in the section “Background & Related Work”. To respect the work that has been done by one of the partners in FIESTA-IoT, the designer of the tool LDoW-PaN[19], we keep this section but move it to this appendix, since all other tools were not fully described in detail in the same way. LDoW-PaN is a Linked Data presentation and navigation model, based on the Dexter Model[20], focused on the average user (see Figure 19).

The purpose of the analytical filtering layer is to scan RDF descriptions to determine triples that contain irrelevant information (from the perspective of the end user) that can therefore be removed, such as meta triples and redundant triples. The analytical filter greatly reduces the number of triples to be treated in other layers from the sequence in Figure 19.

After performing data processing, the analytical filtering layer produces a refined model as output. Although manipulated, the model still complies with the RDF standard. To obtain the benefits offered by the Dexter execution layer, the RDF description that results from the analytical filtering layer must be mapped to the Dexter model. Therefore, the objective of the mapping layer is to translate structured data in the RDF standard to data that meet the characteristics of the Dexter model. After this mapping process, the result is a hypertext network (adapted to the needs of the LDoW-PaN model) modelled according to Dexter's standards, which will be used for subsequent layers in the sequence of activities. The Dexter network is important especially at the interface preparation and presentation and navigation layers, in which characteristics of the Dexter model and its runtime layer are used to create specialized structures for facilitating the presentation and navigation task to be focused on the average user.

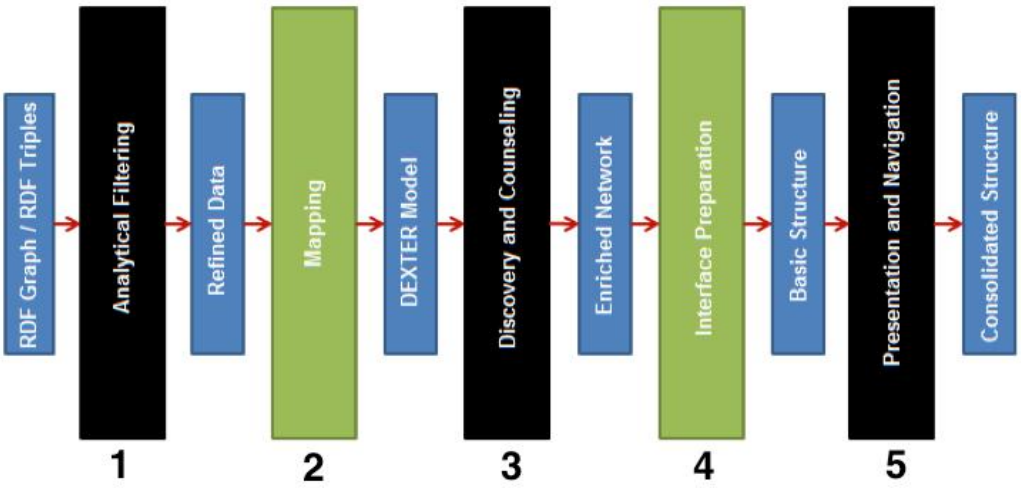


Figure 19: LDoW-PaN model

After the work performed by the mapping layer, the Dexter network is ready to be used by subsequent layers. Most often, the network produced by the mapping

[19] https://github.com/WiserUFBA/LDoW-PaN/wiki

[20] https://github.com/dexter/dexter

process has a considerable amount of nodes and links. To facilitate the work of the discovery and counselling layer and the remaining layers of the LDoW-PaN model, the most relevant network atom must be chosen. This choice has the purpose of creating a target at which the layers can direct their efforts, thereby reducing the computational cost and accelerating the process. For example, the discovery and counselling process involves external access to different data sources and domains. In this scenario, it is possible to imagine an attempt to obtain new knowledge for each atom in the newly created network. Depending on the amount of nodes in the network, the process would easily become unfeasible in terms of performance in responses. Therefore, from this point on, the work performed by layers is centered on a single atom (the most relevant one).

Figure 20 illustrates steps of the discovery and counselling process. In step 1, a basic SPARQL query is executed on the data source that contains the queried resource, which is represented by the most relevant atom of the Dexter network, via its URI. The result of this query is the RDF description of the resource. In this description, among other things, *sameAs* links can be identified. Thus, in step 2, queries are executed based on these *sameAs* links. The results are new RDF descriptions of the queried resource, containing relationships with other resources and literals that were unknown before. Subsequently, in step 3, queries are executed with consideration of the relationships that define the types of the queried resource (*rdfs:type* links). Queries by type return resources that are similar to the queried resource. These resources may not necessarily be linked to the queried resource. In step 4, the final step, a query to the Web 2.0 APIs is executed to obtain elements that are related to the queried resource, such as videos, photos and news, among others.

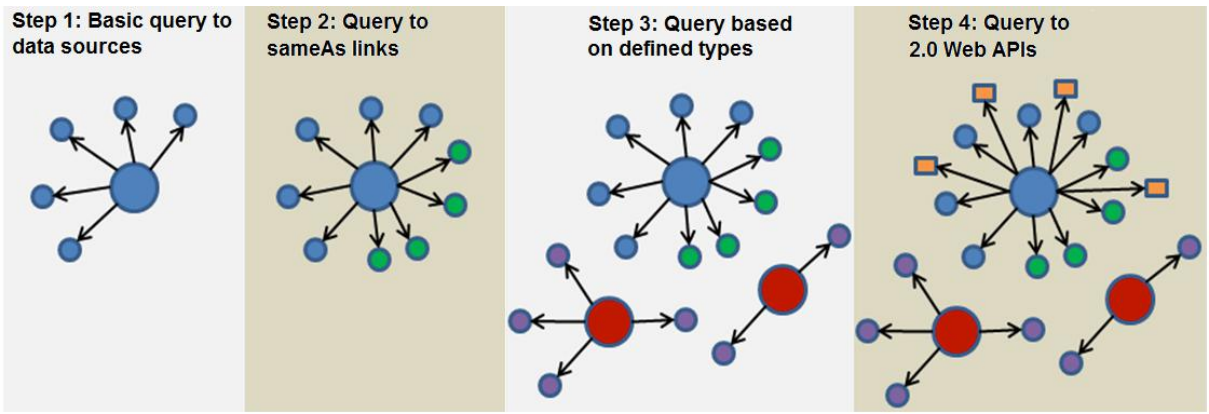


Figure 20: Step-by-step description of the discovery and counselling process

The purpose of the interface preparation layer is to create elements that can be used to facilitate the construction of interfaces, especially those concerned with Linked Data presentation and navigation and focused on inexperienced users. The interface preparation layer provides structures that are ready to be used by other higher-level layers (for example, the presentation and navigation layer) or directly by systems that want to implement interfaces based on these structures. The structures created in the interface preparation layer can refer to the most relevant atom (automated constructions) or a specific atom, which is defined by an action (a click, for example). Four of the structures created on the interface preparation layers (Figure 21):

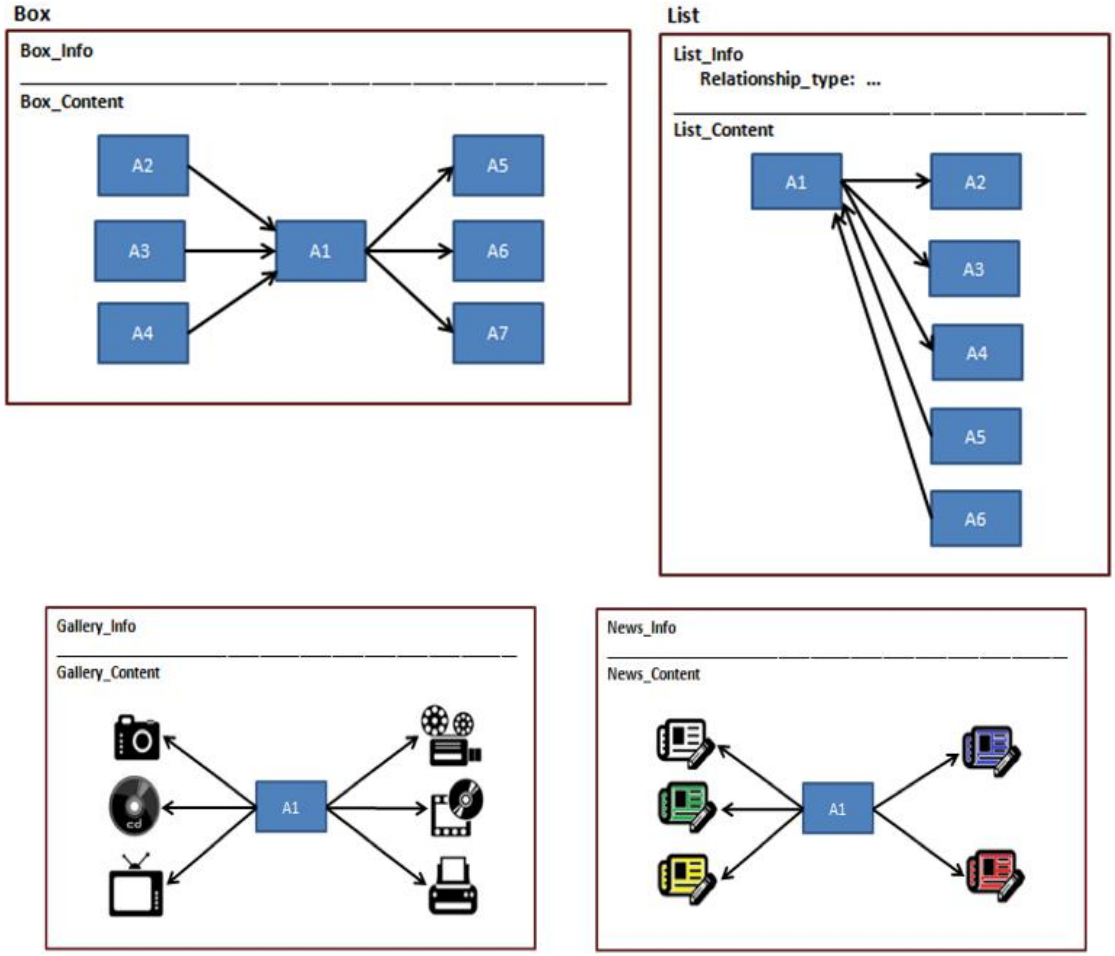


Figure 21: Box, List, Gallery and News structures of LDoW-PaN model.

The presentation and navigation layer uses structures provided by the interface preparation layer to effectively present and provide navigation through data contained in the Dexter network produced during the process. The presentation and navigation layer is the highest-level layer of the LDoW-PaN model, and it implements the interface between the user and the network. A Box structure and a corresponding interface, which can be generated from this structure (Figure 22):

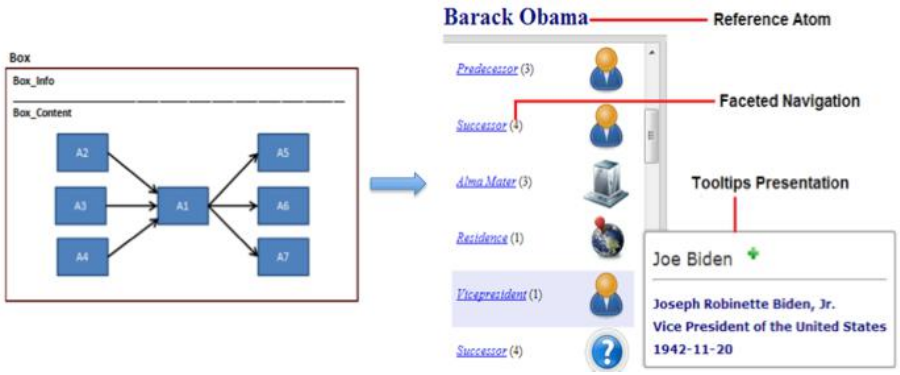


Figure 22: Interface built from a Box

The interface preparation and presentation and navigation layers are extensible. We can create as new structures as new presentation and navigation ways. It can be

adjusted to create more robust and adequate interfaces according to a system's needs (IoT Data, for example Figure 23):

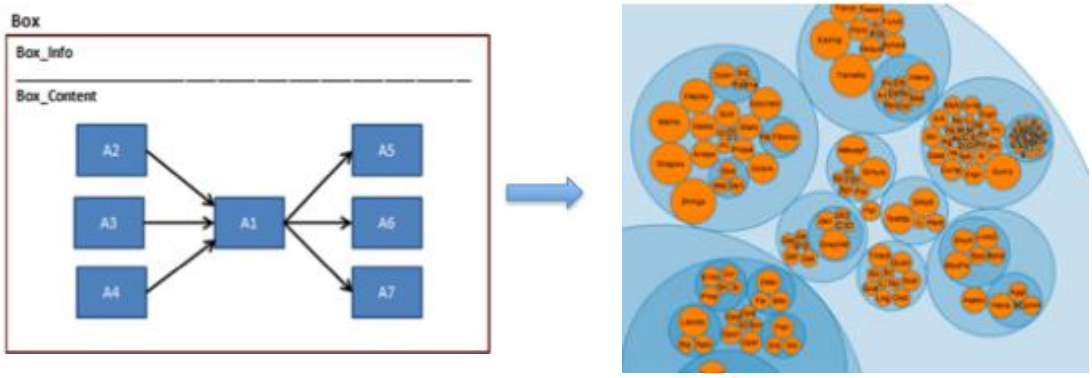


Figure 23: Interface built from a Box (2)

Actually, the LDoW-PaN model is implemented as following: i) a Web Service, which implements the four lower-level layers of the LDoW-PaN model; ii) a client-side script library, which implements the presentation and navigation layer; and iii) a browser extension, which uses these tools to provide Linked Data presentation and navigation for users browsing the Web.

APPENDIX II – VISUALIZING ONTOLOGIES USING WEBVOWL

This section presents a proof-of-concept work carried out using the WebVOWL tool to visualize ontologies to encourage the use of ontologies or an easy interaction for a first approach to learn the ontology. The WebVOWL is easy-to-use, it requires the URI of the ontologies.

The entire graph is brows-able see Figure 24. The experimenters can click on the FIESTA-IoT picture, and the ontology visualization will be automatically displayed. Experimenters can also visualize other ontologies that have been used within FIESTA-IoT such as SSN ontology, IoT-lite ontology and M3-lite taxonomy with the same tool.

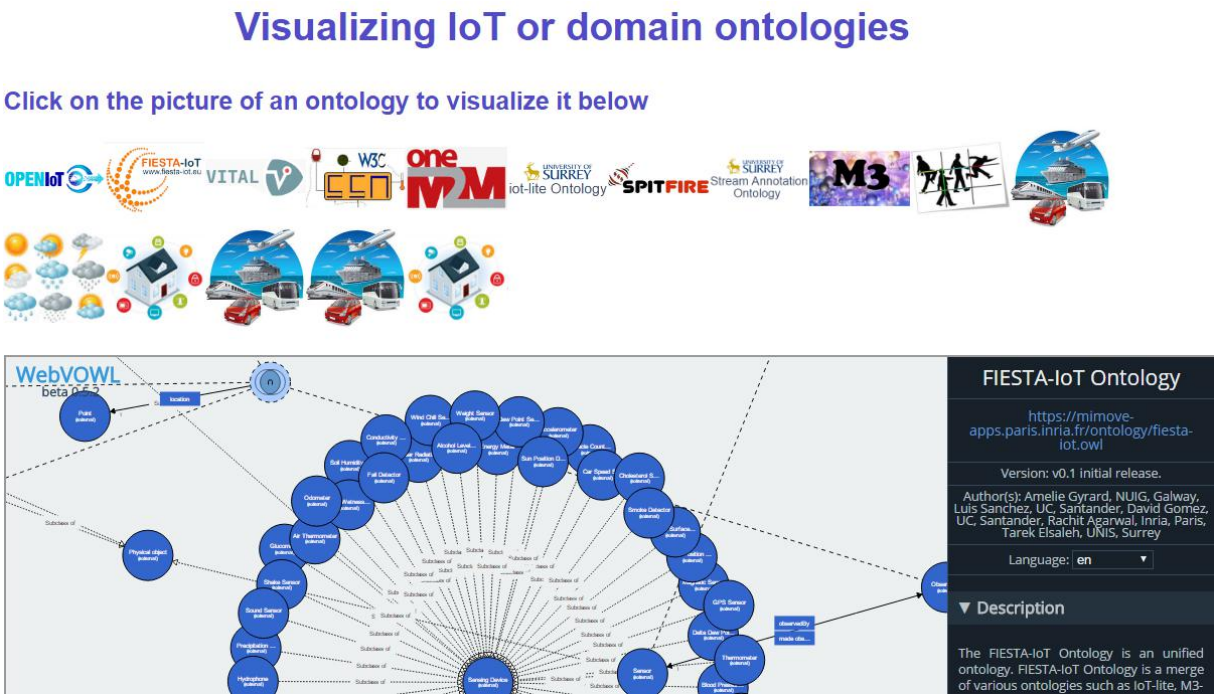


Figure 24. Visualizing IoT ontologies: FIESTA-IoT, IoT-lite and M3-lite

APPENDIX III – DISCOVERY OF TESTBEDS: PROOF-OF-CONCEPT

In this appendix, we provide an example mechanism for the discovery of testbeds already registered within the FIESTA-IoT Meta-Cloud.

Overview

This was an initial work carried-out solely for the purpose of exercise to fulfil the FIESTA-IoT requirements. Furthermore, its uses the initial versions of FIESTA-IoT ontology, SPARQL endpoint, and other work related to the platform. This work discusses providing a web service to query the registered testbeds within the FIESTA-IoT Meta-Cloud.**Error! Reference source not found.** Figure 25 below explains the sequence diagram regarding the testbed discovery process. Users access through the GUI or directly using web services to query all registered testbeds. The GetAllTestbed web service will execute a SPARQL request to query the testbed RDF dataset referencing testbeds registered within the FIESTA-IoT platform. The mechanism is modular and generic enough to work with different SPARQL queries.

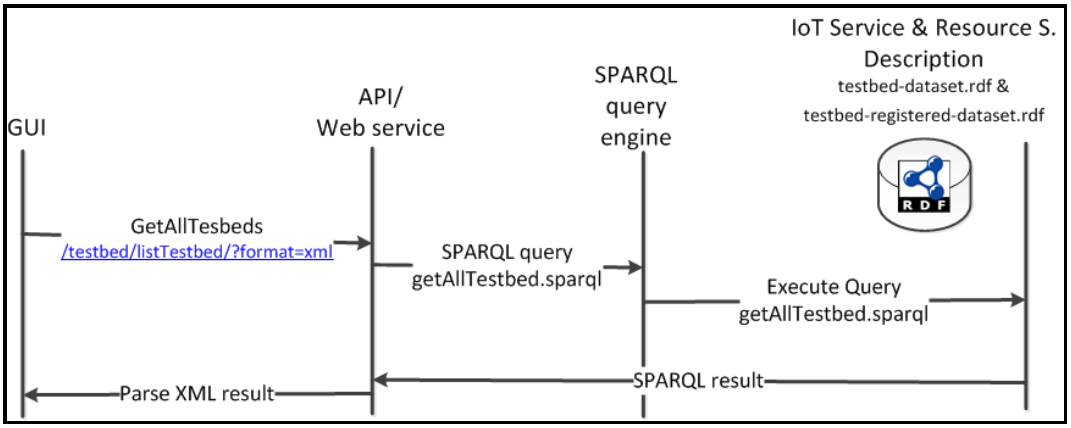


Figure 25: Testbed & Resource discovery sequence diagram

Graphical User Interface (GUI)

Error! Reference source not found. Figure 26 below shows the user interface using the web service /testbed/listTestbed/?format=xml. The drop-down list is filled with the result returned by the web service.

Figure 26: The /testbed/listTestbed/ web service is used within the GUI

Using the same mechanism, with another SPARQL query, we can access to the list of services provided by testbeds already registered within FIESTA-IoT, as depicted in figure below**Error! Reference source not found.**. Adding a new API is really simple, there is just the need of creating a new SPARQL query in the back end of the modular and flexible FIESTA-IoT platform (Figure 27).

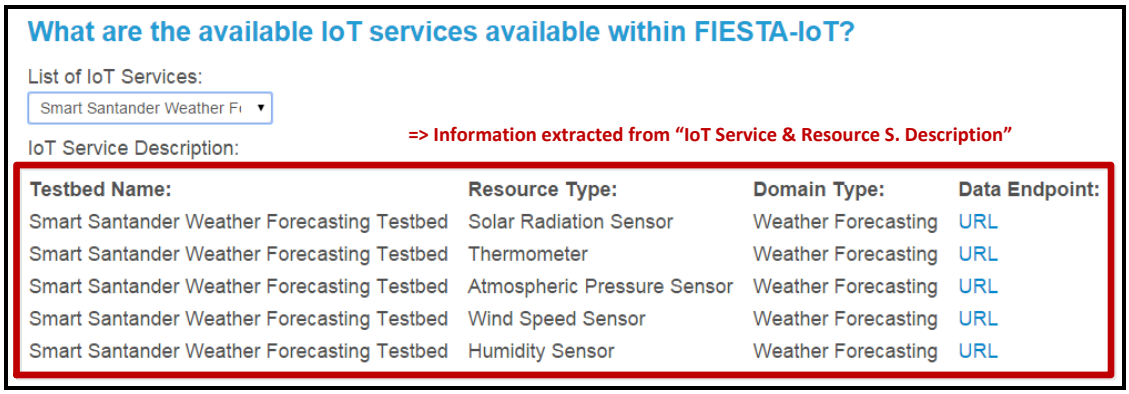


Figure 27: Display the description of an IoT Service

Querying the list of all testbeds with the web service /testbed/listTestbed/

For instance, enter the following URL in a web browser:

http://fiesta-iot-tools.appspot.com/testbed/listTestbed/?format=xml The real platform is under development: http://platform.fiesta-iot.eu/

or

http://fiesta-iot-tools.appspot.com/testbed/listTestbed/?format=json The real platform is under development: http://platform.fiesta-iot.eu/

The result returns a list of testbeds with four elements as depicted in Figure 28: **Error! Reference source not found.**

- **testbedURL** is the URL of the testbed.
- **datasetURL** is the URL to get access to the data produced by devices/resources.
- **testbedName** is the name of the testbed (e.g., Smart Santander Weather Forecasting Testbed)
- **testbedDescription** is the description of the testbed (e.g., The SmartSantander testbed measures temperature (in DegC), humidity (in percent), pressure, wind speed and solar radiation.)

The result format is a ResultSet provided by the Jena framework, more precisely by the Jena ARQ query engine[21].

[21] https://jena.apache.org/documentation/query/

Figure 28: XML result returned by the listTestbed web service

Figure 29 **Error! Reference source not found.**includes a similar web service request returning the result in JSON format.

Figure 29: JSON result returned by the listTestbed web service

SPARQL query

Figure 30 below**Error! Reference source not found.** shows the SPARQL query to return the available services provided by the testbeds available within the FIESTA-IoT Meta-Cloud.

Figure 30: GetAllTestbed SPARQL query

IoT Service & Resource S. Description Repository

IoT Service & Resource S. Description is a RDF repository using the ontologies mentioned in WP3, Task 3.1.

The figure below**Error! Reference source not found.** shows the RDF description of the “IoT Service & Resource S. Description” repository of two testbeds:

- Insight Testbed comprises five resources: Thermometer, HumiditySensor, AtmosphericPressureSensor, CO2_Sensor and SoundSensor. The descriptions of the resources are based on the M3-lite taxonomy.
- SmartSantander Weather Testbed comprises five resources: Thermometer, HumiditySensor, AtmosphericPressureSensor, WindSpeedSensor and SolarRadiationSensor. The descriptions of the resources are based on the M3-lite taxonomy.

Error! Reference source not found. The Figure 31 shows a testbed description semantically annotated with RDF and the following ontologies and taxonomies:

- W3C Semantic Sensor Networks (SSN) to describe which devices are used within the testbed. The property ssn:hasSubsystem is used for this task.
- IoT-lite is used to described the endpoint to get access to the data through the iot-lite:endpoint property.
- M3-lite is used to describe devices in a unified way (e.g., Thermometer) or applicative domains (e.g., Weather). To link the testbed description to the applicative domain, we used the ssn:featureOfInterest property.

testbed-dataset.rdf

```
<ssn:System rdf:about="&iot-lite;InsightTestbed">
   <ssn:featureOfInterest rdf:resource="&m3-lite;Weather"/>
   <rdfs:label xml:lang="en">Insight Smart Living Lab Testbed</rdfs:label>
   <ssn:hasSubsystem rdf:resource="&m3-lite;Thermometer"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;HumiditySensor"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;AtmosphericPressureSensor"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;CO2_Sensor"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;SoundSensor"/>
   <rdfs:comment xml:lang="en">The Insight testbed called Smart Living Lab measures temperature (
   <iot-lite:endpoint rdf:resource="https://www.netatmo.com/addguest/index/RdYu3VJ9HQPqO/70:ee:50
</ssn:System>

<ssn:System rdf:about="&iot-lite;SmartSantanderWeather">
   <ssn:featureOfInterest rdf:resource="&m3-lite;Weather"/>
   <rdfs:label xml:lang="en">Smart Santander Weather Forecasting Testbed</rdfs:label>
   <ssn:hasSubsystem rdf:resource="&m3-lite;Thermometer"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;HumiditySensor"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;AtmosphericPressureSensor"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;WindSpeedSensor"/>
   <ssn:hasSubsystem rdf:resource="&m3-lite;SolarRadiationSensor"/>
   <rdfs:comment xml:lang="en">The SmartSantander testbed measures temperature (in DegC) </rdfs:co
   <iot-lite:endpoint rdf:resource="http://smartsantander.eu/wiki/index.php/Testbeds/Santander"/>
</ssn:System>
```

Figure 31: IoT Service & Resource S. Description RDF Repository

