SEMANTIC HYPERCAT

MICHALIS GEORGIOU

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Master of Science by Research

January 2019

Abstract

The rapidly increasing number of sensor networks and smart devices contributed to the generation of a huge number of information. Information that is generated by several sources and is available in different formats highlights interoperability as one of the key preconditions for the success of the Internet of Things (IoT). Hypercat is a specification defining a JSON-based catalogue, designed to serve the needs of the industry. In this thesis, I extend the existing work on semantic enrichment of Hypercat by defining a JSON-LD based catalogue. The proposed JSON-LD specification offers a mapping mechanism among JSON and JSON-LD catalogues, while highlighting the fact that JSON-LD could be seamlessly adopted by the Hypercat community.

Acknowledgements

First of all, I would like to thank my supervisor Professor Grigoris Antoniou of the School of Computing and Engineering at the University of Huddersfield. His office was always open for me when I needed him or had a question regarding my research or writing. Professor Antoniou was always happy to discuss with me anything about my master or my daily life outside the University.

Furthermore, I would like to express my gratitude to my co-supervisor Dr. Ilias Tachmazidis for introducing me to the topic from the beginning of my Master and helping me every time I run into trouble with the programming part of my master. Without his help this Master, and thesis would not have been possible.

I would also like to thank both of my above supervisors who gave me the opportunity to write with them a research paper for the MIWAI Conference 2018.

In the end, I want to thank my parents, my grandparents and my girlfriend for providing me with constant support and encouragement throughout this year and all of my years of study.

Thank you.

Author

Michalis Georgiou

Table of Contents

Abstract 1
Acknowledgements 2
Chapter 1 5
Introduction 5
Research Contribution 7
Thesis Structure 7
Chapter 2 8
Background 8
Semantic Web 8
BT Hypercat Data Hub 10
Hypercat 3.00 Specification 10
Hypercat Ontology 12
BT Hypercat Ontology 14
JSON-LD - JSON for Linking Data 16
JSON and JSON-LD 18
Linked Open Data (LOD) and Internet of Things (IoT) 18
Linked Open Data (LOD) and Sensors 19
Linked Open Data (LOD) for Smart Cities 20
Summary 21
Chapter 3 22
Related Work 22
Chapter 4 24
Hypercat JSON to Hypercat JSON-LD 24
Hypercat BT JSON-based to JSON-LD based 28
Example from JSON to JSON-LD based catalogue 36
Hypercat JSON-based to JSON-LD based catalogue parser 43
Chapter 5 47
Hypercat JSON-LD Specification 47
Chapter 6 55
Semantic Search 55
Chapter 7 59
Conclusion 59
References 60

List of Figures

Figure 1: The Hypercat Ontology: Retrieved from (Tachmazidis et al, 2016). 14
Figure 2: The BT Hypercat Ontology: Retrieved from (Tachmazidis et al, 2017). 14
Figure 3: JSON-LD Example with @context: Retrieved from (W3C, 2018). 16
Figure 4: Sample JSON-LD document using full IRIs instead of terms: Retrieved from (W3C, 2018). 17
Figure 5: A Linked Data graph: Retrieved from (Lanthaler & Gült, 2012). 17
Figure 6: Linked open data cloud: Retrieved from The Linked Open Data Cloud. (2018). 19
Figure 7:Linking raw data stream to virtual RDF graph: Retrieved from (Phuoc & Hauswirth, 2009) 20
Figure 8: Service Architecture of the RDF Translator: Retrieved from (Stoltz et al, 2013). 22
Figure 9: Semantic Search Implementation's Design 58

List of Tables

Table 1: Hypercat Core properties mapped to existing JSON properties 13

Chapter 1

Introduction

In 2014, eight industry-led projects were sponsored through Innovate UK (the UK's innovation agency) as part of the Internet of Things Ecosystem Demonstrator programme to deliver IoT `clusters', where every cluster is based on a data hub that aggregates and exposes data feeds from numerous sensor devices (Tachmazidis et al, 2016). Addressing interoperability was the main objective of the programme leading to Hypercat, a hypermedia standard which represents and exposes Internet of Things data hub catalogues through web technologies, to increase further the discoverability and interoperability of data (Davies & Fisher, 2015). Hypercat facilitates the combined usage of distributed data repositories (hubs), thus allowing applications to search (query) the hubs' catalogues in machine understandable format, in order to discover and retrieve the needed data. According to Hypercat Specification (Beart, 2016), interoperability is based on similar principles on which the web and linked data are made. Such principles include information accessibility over standard web protocols and types (HTTPS, JSON, REST), the proof of identity of resources over URIs, and the establishment of common, shared semantics for the descriptors of datasets. From this kind of view, Hypercat can be a practical beginning to solve the problems of working on several data sources, combined inside numerous data hubs, over linked-data and web methods. Hypercat includes a lightweight, JSON-based approach based on a technology which is popular to the developers (Tachmazidis et al, 2016). Every Hypercat catalogue has a list and a description on every URI that points a data source, with a set of relation-value pairs (metadata) connected with it. Thus, a given server has the ability to supply a set of semantically annotated resources using Hypercat. In addition, there is a small group of core compulsory metadata relations-value pairs that a Hypercat catalogue should contain, but developers have also the choice to define any group of annotations they need (Beart, 2016). There is also a Hypercat community with free to use tools available in Github with the link “https://hypercatiot.github.io/”. Dealing with the problems of complexity and diversity of IoT data sets emphasised that linked data and Semantic Web technologies as the key to tackle the problem (Antoniou & Harmelen, 2008). The linked data allows the inclusion of data inside a shared, browsable and approachable knowledge graph while letting the data allocated and managed in dissimilar systems, controlled by different contributors (Tachmazidis et al, 2016). The effective usage of linked data technologies in numerous different cases, such as collecting data from different sources to place them together in a generic way, can allow a diversity of applications, with no need of encoding the restrictions of the applications inside the data model. To meliorate the data analysis and the interoperability among systems, the semantic web technologies apply the ability to add important data

models such as ontologies (Lecue et al, 2014). Therefore, it is reasonable to start thinking and start studying of how the Hypercat standard and its specification could be serialised in a semantic language/form and to examine the advantages that could happen from such a materialization, this is the aim of this research to develop a semantically-enriched Hypercat system. Thus, a Hypercat catalogue can be considered as a more demonstrative and descriptive catalogue where data policies and their corresponding data flows (d'Aquin et al, 2014) are provided as machine readable information. Additionally, it would be helpful to investigate on how a Hypercat specification could be parsed and serialized in a semantic language such as JSON-LD. There is also a previous work of a semantically enriched Hypercat catalogue which is developed in RDF, but the Hypercat RDF-based catalogue has not been adopted because is making transition to whole different format. On the other side JSON-LD can be easily adopted by the developers and Hypercat Community because of the similarity of the JSON-LD's syntax and JSON's syntax. Their basic difference is the two keywords *"@context"* and *"@id"*. Specifically, JSON-LD is 100% compatible with JSON and can add the needed semantics to the catalogue, in addition, JSON-LD can support ontologies directly (Su et al, 2015). There is a full explanation of JSON-LD specification to the following section. According to the W3C documentation about JSON-LD syntax, JSON-LD is designed to store Linked Data into databases that support JSON and meanwhile use the JSON libraries or parsers that are currently available. It is also designed to be usable as RDF and use the semantic web technologies like SPARQL (W3C, 2017). As specified by (Sporny, 2014) one of the primary JSON-LD creators, in his blog, he discusses that JSON-LD is based on a technology that most developers use today. There is also another data format which supports semantic web technologies and can be compatible with RDF and Ontology language like JSON-LD, is the Entity Notation (EN) (Su et al, 2015). The idea to serialise and parse Hypercat from JSON to EN will be the same as the previous work of (Tachmazidis et al, 2016) RDF(N-triples) because is also making transition to a different syntax. Throughout the research that has been taken finding of how the Hypercat system can be semantically-enriched, and adopted easily by the developers and community, it is worth take the advantages of JSON-LD and create a Hypercat JSON-LD based catalogue. Therefore, the existing JSON-based catalogues can be replaced by JSON-LD based catalogues with minimum effort and without changing the existing database. Moreover, the current JSON-based Hypercat developers can embrace the change easily and translate their current JSON-based Hypercat catalogues to JSON-LD based catalogues. Consequently, introducing the new Hypercat JSON-LD based catalogue, the Semantic Search can be proposed. With the help of Semantic search, a user can query a semantically enriched Hypercat catalogue through SPARQL-like queries. In this research thesis, I propose a new version of semantically enriched Hypercat catalogue based on JSON-LD as well as the Hypercat specification that can raise the interoperability. The new catalogue's

data is available based on an existing ontology which can be found in the work of (Tachmazidis et al, 2016) providing a mapping mechanism among the current JSON and proposed JSON-LD properties. A systematic translation of an existing Hypercat JSON catalogue into a Hypercat JSON-LD catalogue is presented. In addition, various aspects of the proposed Hypercat JSON-LD specification are studied in comparison to the existing Hypercat JSON specification.

Research Contribution

The aim of this research is to develop a semantically-enriched Hypercat system, and the objectives are to:

1. Establish familiarity with the Hypercat standard (JSON and RDF).
2. Establish familiarity with the JSON-LD.
3. Develop a JSON-LD equivalent for the Hypercat specification.
4. Develop a parser translating Hypercat JSON-based catalogue to Hypercat JSON-LD based catalogue.
5. Design and propose a Semantic Search mechanism.

Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 delivers a background on the Semantic Web technologies, an explanation of BT Hub, it also provides the current Hypercat 3.00 specification, descriptions of the Hypercat Ontology and BT Hypercat ontology, it introduces the JSON-LD, and in the end it provides the LOD (Linked Open Data). Chapter 3 provides the related work, a description of the current converters and why this work can be different. Chapter 4 describes the translation of a JSON-based catalogue to JSON-LD based catalogue, as long with an example of the translation of BT Hypercat JSON-based catalogue with each properties and aspects to a JSON-LD catalogue and a description of the Hypercat JSON-based to Hypercat JSON-LD based catalogue parser. There is also a small sample of how a BT Hypercat JSON based catalogue can be translated into a BT Hypercat JSON-LD based catalogue. At the end of Chapter 4 it can be seen a description of how the parser from a BT Hypercat JSON to BT Hypercat JSON-LD is working. Chapter 5 presents the Hypercat JSON-LD specification, while Chapter 6 describes the Semantic Search. The conclusion is in Chapter 7, following by the references.

Chapter 2

Background

Semantic Web

The need to present Web information to a form that machines can understand and process it, pushed the Web to evolve to the Semantic Web. Nowadays, the web content, is written in HTML programming language, and it can be captured using the search engines which are easy to use and read by humans. Meanwhile, the information which is generated automatically from a database, is displayed without the prototype structure of a database (Antoniou & Harmelen, 2008). Machines encounter problems to understand data which is in format where humans can understand it, such as HTML. Although the Web can help the machines to find data, it does not provide enough help for them to find the right data. The ability of the machines to find the data is based on a keyword match and its alternatives. In contrast with humans, machines do not understand the meaning of the search keywords. There are also several Artificial Intelligence scientists who believe that a way to tackle the problem of machine understanding is the Machine Learning. Nevertheless, this solution is based on statistical analysis and mathematical equations and also Machine Learning can be extremely expensive even to achieve small tasks such as, recognising a picture (Ismail & Shaikh, 2016). The Semantic Web challenges the problem from the Web page side. Assuming that HTML is changed by further suitable and machine-understandable languages, then the Web pages could bring their information content together. Thus, the key idea is to use machine-processable Web information (Antoniou & Harmelen, 2008). JSON format is used to structure the data, and to enable the machines to parse it or generate it (JSON, N/A). Nevertheless, not even this kind of formats can solve the next problem: "the meaning of web content is not machine-accessible" (Tachmazidis et al, 2016). Next, there is a description of the Semantic Web technologies and standards. These technologies can make the Web information more machine-understandable and help to the implementation of intelligent approaches such as reasoning or make the users jobs easier by automate their tasks. Nowadays, the automation of tasks is more significant for the growth of connected devices that are part of the Internet of Things (IoT) (Tachmazidis et al, 2016). The RDF (Resource Description Framework), is a W3C standard that lets the information to be represented and combined from numerous knowledge domains. With this, the Semantic Web develops a Semantic Network with the information inside the network is characterised as directed labelled graphs (RDF graphs). In the graphs there are nodes that represent either a concept or object and can be identified through a Unique Resource Identifier (URI). The information is understandable and usable by both humans and machines because the information is described in interlinked directed labelled graphs thus, they achieve a uniform representation of information. The RDF graphs can be represented and

serialized in the following formats: RDF/XML(.RDF), Turtle(.TTL), N-Triples(.N_T), and Notation-3(.N3). The RDF Schema (RDFS) is the vocabulary which contains the most basic elements who describe the ontologies. Theses ontologies are improving the semantic structure of the RDF graphs. RDFS vocabulary is comprised by classes, subclasses, comments and data-types (Pauwels et al, 2017). Using Semantic Web standards, a developer can define the machine-understandable semantics of meanings of an application domain. Especially, the ontology represents a clear and formal specification of a conceptualization and describes formally a domain discourse. The ontology can be described by the OWL Web Ontology and has the ability to contain definitions of classes of the domain and relations among these classes such as (class hierarchies). The latest version of OWL (Ontology Web Language) which is also a W3C standard is OWL 2 (W3C, 2012). Ontologies can also deliver a common understanding of the domain, but with such a common understanding is essential to differ in the terminology. For example, the zip code of an application might be similar to other application's source code. The next issue is when two programmes might use the similar term but with dissimilar concepts. This kind of issues can be solved by mapping the specific terminology to a common ontology or by defining straight mappings among the ontologies. Nevertheless, ontologies can support semantic interoperability (Antoniou & Harmelen, 2008). Ontologies can be used for the organization and the navigation of Web sites. Most of the Web sites exposed to the page's top levels of a concept hierarchy of terms, and the user can use them to expand the subclasses. In addition, ontologies are usable for meliorating the precision of Web Searchers. The search engines can find pages that mention the specific concept in an ontology rather than gathering all pages that are not certain and where the keywords occur (Antoniou & Harmelen, 2008). The data about the application domain objects and their relationships can be supported in JSON-LD website. Moreover, Web resources represented in OWL could allow for automatically interfering implied facts about these resources. Turtle stands for the 'Terse RDF Triple Language' and is another syntax which represents data in RDF model. Turtle represents RDF graphs in which are made up of triples having subject, predicate and object in a compact form. It also offers short-hand forms of public usage patterns, and different datatypes. Turtle provides similarity with both N-Triples and Notation 3 syntaxes, both are also RDF serializations. It also has similar triple design syntax with SPARQL (W3C, 2014). Another W3C standard is SPARQL, this is a query language which belongs to the category of the Semantic web. SPARQL is almost identical with the SQL (Structured Query Language) for the RDBMS (Relational Database Management System). This query language can be used to query RDF files. The results of SPARQL queries can be a set of graphs because the RDF files are described and portrayed in labelled triples, so this lets a graph representation of RDF files. A query result set can be a literal (value) or a URI. The result set can be used easily and straight by the applications because SPARQL can bring the literal or

transform the URI into their labels (Ismail & Shaikh, 2016). A characteristically hypothesis of the Internet of Things (IoT) scenarios, is the importance of automatic inference and retrieval when there is a huge amount of data that change fast (such as streaming data) (Gubbi et al, 2013).

BT Hypercat Data Hub

BT Hypercat Data Hub is a platform which makes accessible the information from a large range of sources and presents them to the users or developers in a reliable way. The data hub's portal delivers an interface to the user who can use the specific data, where they can browse a data catalogue and then choose or subscribe to the data feeds they need. Additionally, both of JSON-based and RDF-based Hypercat catalogues are offered, with the RDF-based catalogue proposed and defined by (Tachmazidis et al, 2016). An API authorizes admission to data feeds, protected by API keys, whilst a relational, GIS capable, database allows complicated queries that information can be filtered based on a huge number of standards. A range of last technology sensors allows the data to insert into the data hub and to be converted to a typical format for usage into the platform's core. The data hub offers a steady method for inclusion among information retrieved by devices, systems, and people through transmission networks and the software that may need this information to advance decision making, such as, in control systems. The group of sensors which are included in the hub are for input and output. These sensors are used for specified information resource or application feed and might be carried out in different cases. The data among arbitrary exterior formats and the interior data formats must be decoded. While using web technologies, the Hypercat standard, is able to represent and expose Internet of Things data hub catalogues, to boost the data interoperability and discoverability. Of course, the main idea is to allow dispersed data repositories (data hubs) to be used combined by several programmes and make possible to search into their catalogues in a machine-understandable format. This allows the build of knowledge graphs of the datasets via numerous hubs that specified applications query and access the wanted information (data), this can be done in any datahub in which they are maintained. Considering this, Hypercat represents the beginning of solving the problems of controlling numerous data sources, aggregated into numerous data hubs, thru semantic web methods and linked data (Tachmazidis et al, 2017).

Hypercat 3.00 Specification

In this subsection, it can be found the basic description of the Hypercat 3.00 specification discussed by (Tachmazidis et al, 2016). Hypercat is a lightweight JSON-based hypermedia catalogue format for presenting large groups of URIs, with every URI carrying numerous RDF-like triple statements about it. The Hypercat catalogue is a document which exposes an unclassified collection of information on

the web, with the help of its URI, every catalogue item points out to a unique resource. Therefore, each Hypercat catalogue can reveal a list of data like data feeds and deliver different links (URIs) to other Hypercat catalogues. However, it is not permitted to define a catalogue inside a catalogue, so catalogues can be linked by mentioning to other catalogue URIs. Additionally, a catalogue can supply meta-data about every catalogue item and itself (Tachmazidis et al, 2016). The *Catalogue object* which is a JSON object, is the base of the Hypercat catalogue's structure. The Catalogue object must include the next parameters:

i. "*items*", which is the list of items with a property value of a "*JSON array containing zero or more Item Objects*".
ii. "*catalogue-metadata*", represents an array of Metadata Objects defining the catalogue object with the property value of a "*JSON array of Metadata Objects*".

A single *Item Object* which comes from the items array, is a JSON object and must include the next properties:

i. "*href*", the identifier of the resource item with a property value of a "*URI as a JSON string*".
ii. "*item-metadata*", a single array of Metadata Objects which outlines the resource item with a property value of a "*JSON array of Metadata Objects*".

The arrays, *catalogue-metadata* and *item-metadata* must include at least one metadata object for every of the mandatory Metadata Object relationships. Both of the aforementioned arrays may include several metadata objects with the same "*rel*" and "*val*" properties (multisets of features). The *Metadata Object* is a JSON object and represents a single relationship among the source object (whether is catalogue or catalogue item), and some further entity or concept declared by a URI, this kind of relationship is practicable to every catalogue item to the catalogue itself. A *Metadata Object* has the next properties:

i. "*rel*", is the relationship among the source object and a focus noun, declared as a predicate with the property value of a "*URI of a relationship as a JSON string*".
ii. "*val*", is the object (noun) to which the *rel* property applies with the property value of a "*JSON string, it can also be the URI of a concept or entity*".

There are also two compulsory metadata relationships which must be included in all catalogues such as:

i. "*rel*": "*urn:X-hypercat:rels:hasDescription:en*" meaning that the resource has human readable explanation in English language ("*Description as a JSON string*").

ii. "*rel*": "*urn:X-hypercat:rels:isContentType*" which means that the data which is provided by resource is of the given MIME type (the "*val*" of this relationship: "*application/vnd.hypercat.catalogue+json*").

The second relationship (ii) must be included in every top-level of the *catalogue-metadata* objects (describing the catalogue object) (Tachmazidis et al, 2016). The above Hypercat structure which has been described, composes only the basic core of any given Hypercat catalogue. Yet, the whole Hypercat 3.00 Specification supplies a full description of the model related to the above explanation. Thus, in the next chapters of this thesis, there is a definition of every aspect of the Hypercat 3.00 specification while giving a commensurable semantically enriched service supported by an OWL ontology, which is stated in RDF.

Hypercat Ontology

In this subsection, there is a definition of the Hypercat ontology (Tachmazidis et al, 2016), which captures the above-mentioned Hypercat structure, therefore delivering a translation mechanism from JSON-based catalogue to a JSON-LD based. The Hypercat ontology is accessible with the uri:

"http://portal.bt-hypercat.com/ontologies/hypercat"

and includes the main properties that would make possible the development of JSON-LD based catalogue. Each namespace of the Hypercat ontology must be written by prefixing concepts and properties with "*hypercat:*", this is why the uri look like this "http://portal.bt-hypercat.com/ontologies/hypercat". At the moment, this ontology is only part of an IoT Data Hub "https://portal.bt-hypercat.com/", whilst as a future plan, there will be a suggestion to the Hypercat community to standardise it. Having semantically enriched catalogues such as JSON-LD or RDF, and supported by a well-designed ontology, it believed that it would boost up the interoperability, offer intelligence reasoning capabilities and implement structured data to the catalogue (Tachmazidis et al, 2016). Figure 1 portrays the class hierarchy which is used to create the Hypercat Ontology, and a range of properties it is included in Table 1 (Tachmazidis et al, 2016). The Table 1 which retrieved from (Tachmazidis et al, 2016), in the second column demonstrates the properties as RDF-based but the same properties can be used to the JSON-LD based translation. The core hierarchy collects all of the JSON-based catalogue structure and at the same time it can provide pliability for additional extensions. The Hypercat ontology consists of class *MetadataAnnotator* (having two subclasses, *Catalogue* and *Items*) and class *Search*. It can be seen below that; a Metadata Object is practicable to the catalogue itself and every catalogue item. It can be seen that; the class *Metadata Object* can be applied to every

catalogue item and to the catalogue itself. Note that (Tachmazidis et al,2016), defined the new class *MetadataAnnotator*, this class can capture every metadata property which is practicable to both of *Item Objects* and *Catalogue Objects*. Hypercat 3.00 Specification describes that there are some properties that are only practicable to the one of the following: *Item Objects* or *Catalogue Objects*. This kind of properties are not part of the class *MetadataAnnotator's* definition. Afterwards, there are two classes, the class *Catalogue* which models a *Catalogue Object* describing properties that are practicable only to catalogue's metadata, and the class *Item* which models the *Item Object* describing properties that are only practicable to an item's metadata. Both of the aforementioned classes are subclasses of the class *MetadataAnnotator*. However, to be able to develop a semantically enriched catalogue, the property *hasItem* must be added on top of the catalogue to show the connection between class *Catalogue* and class *Item*. This can also show that a *Catalogue* contains a list of *Items*. Additionally, the class *Search* models the numerous search types that can be available from the *Catalogue*, and the property *supportsSearch* can connect these two classes (Tachmazidis et al,2016). The collection of the available searcher types is described over individuals of class *Search*. A more detailed description of the available search types will be explained to Chapter 4.

JSON-based	**RDF-based**
urn:X-hypercat:rels:hasDescription:en	https://www.w3.org/2000/01/rdf-schema#comment
urn:X-hypercat:rels:supportsSearch	hypercat:supportsSearch
urn:X-hypercat:rels:isContentType	hypercat:isContentType
urn:X-hypercat:rels:hasHomepage	hypercat:hasHomepage
urn:X-hypercat:rels:containsContentType	hypercat:containsContentType
urn:X-hypercat:rels:hasLicense	hypercat:hasLicense
urn:X-hypercat:rels:acquireCredential	hypercat:acquireCredential
urn:X-hypercat:rels:eventSource	hypercat:eventSource
urn:X-hypercat:rels:hasRobotstxt	hypercat:hasRobotstxt
urn:X-hypercat:rels:accessHint	hypercat:accessHint
urn:X-hypercat:rels:lastUpdated	hypercat:lastUpdated
urn:X-hypercat:search:simple	hypercat:SimpleSearch
urn:X-hypercat:search:geobound	hypercat:GeoboundSearch
urn:X-hypercat:search:lexrange	hypercat:LexrangeSearch
urn:X-hypercat:search:multi	hypercat:MultiSearch
urn:X-hypercat:search:prefix	hypercat:PrefixSearch
urn:X-hypercat:search:semantic	hypercat:SemanticSearch

Table 1: Hypercat Core properties mapped to existing JSON properties: Retrieved from (Tachmazidis et al, 2016).

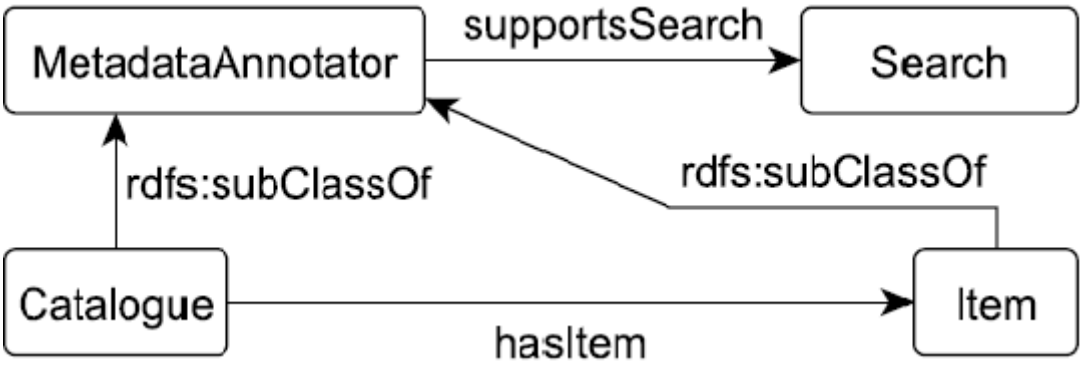

Figure 1: The Hypercat Ontology: Retrieved from (Tachmazidis et al, 2016).

BT Hypercat Ontology

BT Hypercat Ontology is an ontology which extends the core of the Hypercat specification. Thus, this is an example of how the ontologies extent to the Hypercat core ontology and specification. This ontology will be also an example to the translator for JSON-LD based Hypercat catalogue to the following chapter. In the BT Hypercat Data Hub, the data can be enriched by representing the information in RDF using properties and concepts that are well-defined in an OWL ontology (Tachmazidis et al, 2017).

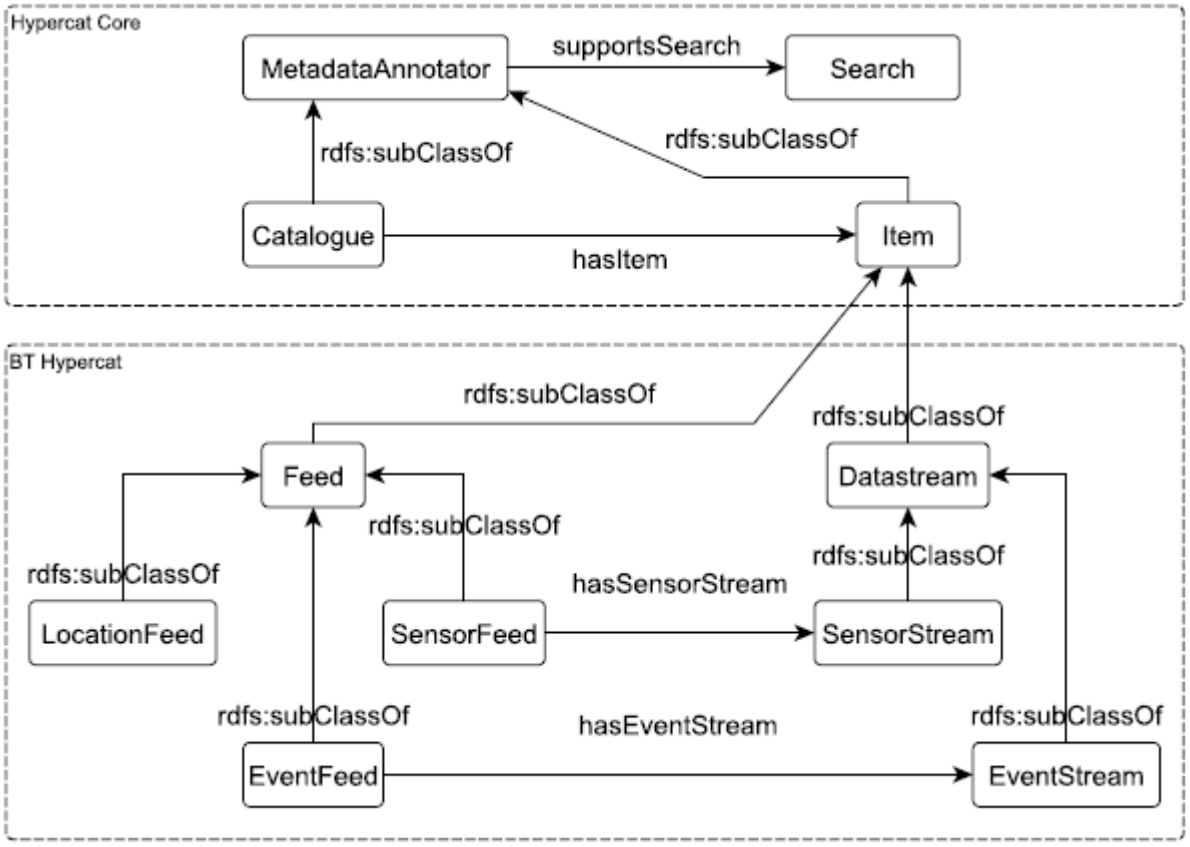

Figure 2: The BT Hypercat Ontology: Retrieved from (Tachmazidis et al, 2017).

In Figure 2 there is a demonstration of the *BT Hypercat Ontology*. It can be seen that; the *Feed* is the main class for every data feed that is declared in the knowledge base. The class *Feed* has the semantic properties of feeds such as, feed_tag, feed_id, feed_creator, feed_email, feed_title, feed_status, feed_description, feed_location_name and feed_domain (Tachmazidis et al, 2017). Moreover, there are also three subclasses of class *Feed*:

i. "*SensorFeed*" subclass which represents the feeds.
ii. "*EventFeed*" subclass which represents the events.
iii. "*LocationFeed*" subclass which represents the locations.

The modelled data has been incorporated into the BT Hypercat Data Hub and it can be one of the next feed types:

i. "*SensorFeed*"
ii. "*EventFeed*"
iii. "*LocationFeed*"

In brief, every data source can present their available information over the *BT Hypercat Data Hub* by delivering a single feed. A feed is a resource coming either from sensors, events or locations. Inside every feed, there is data which is available over datastreams, and it is defined by the class *Datastream.* The class *Datastream* has two subclasses:

i. "*SensorStream*" subclass which represents datastreams from sensors.
ii. "*EventStream*" subclass which represents datastreams from events.

Therefore, a feed can also deliver a set of datastreams that are related to each other, for example a data feed which provides information about weather, can also deliver datastreams for temperature, pressure, or wind speed. Information for the locations can be provided by a feed type of "*LocationFeed*".

The BT Hypercat Ontology has been created and is available online with the uri:

"http://portal.bt-hypercat.com/ontologies/bt-hypercat"

There are also BT Hypercat online catalogues both in JSON and RDF (a work by Tachmazidis et al, 2016-2017). These catalogues are available with the uris "*http://portal.bt-hypercat.com/cat*" and "*http://portal.bt-hypercat.com/cat-rdf* " respectively. These catalogues contain details about the feeds and information sources sideways with extra metadata like tags, which let better-quality search and discovery (Tachmazidis et al, 2017).

JSON-LD - JSON for Linking Data

JSON-LD is attempting to create a simple technique to express Linked Data in JSON and at the same time enhance semantics to JSON files. The designing of JSON-LD is very simple and human understanding. The main goal was to be very easy for the developers to understand it and translate their JSON documents to semantically JSON-LD. Thereafter, a triple-centric approach was followed, and it was looking like a straight conversion from Turtle to JSON. For this reason, to the latest versions of JSON-LD, the syntax was transformed and allows the serialization of data to be more indiscernible, from traditional JSON. The only thing that a developer needs to know to develop a simple JSON-LD document is plain JSON and two keywords which are "*@context*" and "*@id*" to use the basic functionality of JSON-LD. The "*@context*" is the keyword which is used to describe briefly the names that are part of the JSON-LD file. This kind of small names are the terms, and the developers used them to express detailed identifiers in a compact manner. The keyword "*@id*" identifies the unique node objects that are being defined in the file with IRIs (W3C, 2018). In the Figure 3 there is an example of a JSON-LD document.

```
{
  "@context": {
    "name": "http://schema.org/name",  ← This means that 'name' is shorthand for
'http://schema.org/name'
    "image": {
      "@id": "http://schema.org/image",  ← This means that 'image' is shorthand
for 'http://schema.org/image'
      "@type": "@id"  ← This means that a string value associated with 'image'
should be interpreted as an identifier that is an IRI
    },
    "homepage": {
      "@id": "http://schema.org/url",  ← This means that 'homepage' is shorthand
for 'http://schema.org/url'
      "@type": "@id"  ← This means that a string value associated with
'homepage' should be interpreted as an identifier that is an IRI
    }
  }
}
```

Figure 3: JSON-LD Example with @context: Retrieved from (W3C, 2018).

Next in the Figure 4 there is an example of a JSON-LD document without the "@context" keyword.

```
{
  "http://schema.org/name": "Manu Sporny",
  "http://schema.org/url": { "@id": "http://manu.sporny.org/" },  ← The '@id'
keyword means 'This value is an identifier that is an IRI'
  "http://schema.org/image": { "@id": "http://manu.sporny.org/images/manu.png" }
}
```

Figure 4: Sample JSON-LD document using full IRIs instead of terms: Retrieved from (W3C, 2018).

While JSON-LD is 100% compatible with plain JSON, can let the developers use their current libraries and tools. This can also be very helpful and important for businesses because it let them enhance the meaning of their existing JSON files without causing troubles to them or to their customers. Meanwhile, JSON-LD can support every major RDF concept (Lanthaler & Gült, 2012).

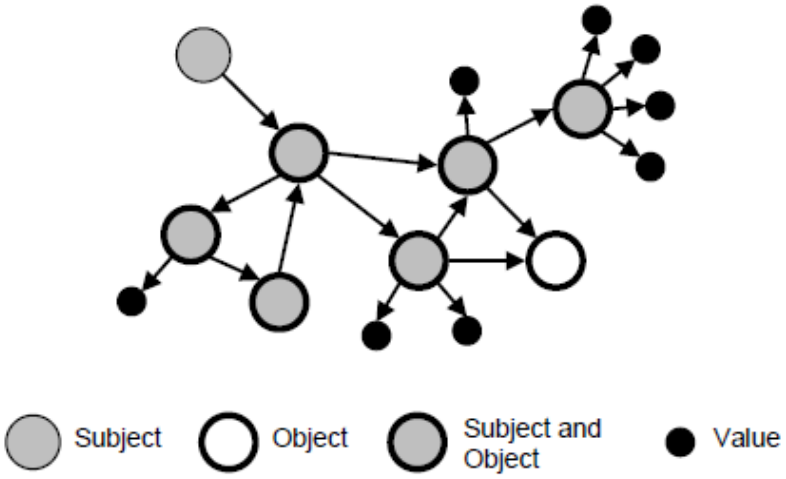

Figure 5: A Linked Data graph: Retrieved from (Lanthaler & Gült, 2012).

Figure 5 demonstrates a JSON-LD's data model, which is a Linked Data graph. It can be seen that the nodes inside the data graph are named *subjects* and *objects* while the edges are named *properties* (predicates in RDF). The node which has at least one leaving edge is the *subject* while the *object* is the node which has at least one received edge. This means that a node can be a *subject* and an *object* simultaneously. A *subject* to be identified and referenced easily should be labelled with an IRI. Fortunately, this requirement is not mandatory because JSON-LD can support unlabelled nodes too. Although, unlabelled nodes, do not meet the demands of Linked Data, they are supported because they

let specific use cases, which need only local data. Again, the same goes for the *properties* (edges): if they are labelled by IRI then they are referenceable and makes them Linked Data; if not they are just JSON *properties* that have only sense in the particular file. On the other hand, for the *objects* is different, if the *object* is labelled by an IRI, then it is called *object*, but if it is labelled by something else that is not IRI such as numbers, it is called *value*, for example, RDF's literal (Lanthaler & Gült, 2012).

JSON and JSON-LD

As it mentioned above, JSON (JavaScript Object Notation) format is used to structure the data, and to enable the machines to parse it or generate it. Moreover, is very simple and it derives from JavaScript. JSON can be built on two structures, the first one is, pairs of value and name and the second one is, a list of values (arrays) (JSON, N/A). JSON does not have the ability to support native hypermedia, namespaces or semantic annotations. The JSON Schema has been proposed as an example to tackle this problem and add hypermedia support to the JSON through the *'$ref'* keyword to represent a hyperlink. In addition, more proposals have been proposed to solve the JSON's lack of abilities above, ending to the JSON-LD project. The key ability of JSON-LD is to give semantic meanings to the JSON files (Lanthaler & Gült, 2012). Both JSON and JSON-LD serializations are the same syntactically and the only difference is the JSON-LD's keywords (Bradley, 2014).

Linked Open Data (LOD) and Internet of Things (IoT)

The Hypercat can be the key for delivering semantically enriched data, this can be extended by permitting the specific identification (unique ID) of the current source. The interoperability over several domains and the extra enrichment can be allowed by merging the local stored data with the Linked Open Data cloud (LOD) (Tachmanzidis et al, 2017). The Linked Open Data (LOD) is a web approach of the Semantic Web to connect data that are accessible through several dissimilar sources. Inside this cloud can be found structured and unstructured sources both letting to the rise of an adaptable model of data utilization. In this crowd source plan (LOD) there are over 31 billion facts in the cloud since 2014 (Ismail & Shaikh, 2016).

The Figure 6 represents the Linked Open Data Cloud diagram, and the dataset has 1,231 datasets with 16,132 links since June 2018 (The Linked Open Data Cloud, 2018). According to (Tachmanzidis et al, 2017) the LOD can be used to enable federated queries over the BT SPARQL Endpoint they present in their work.

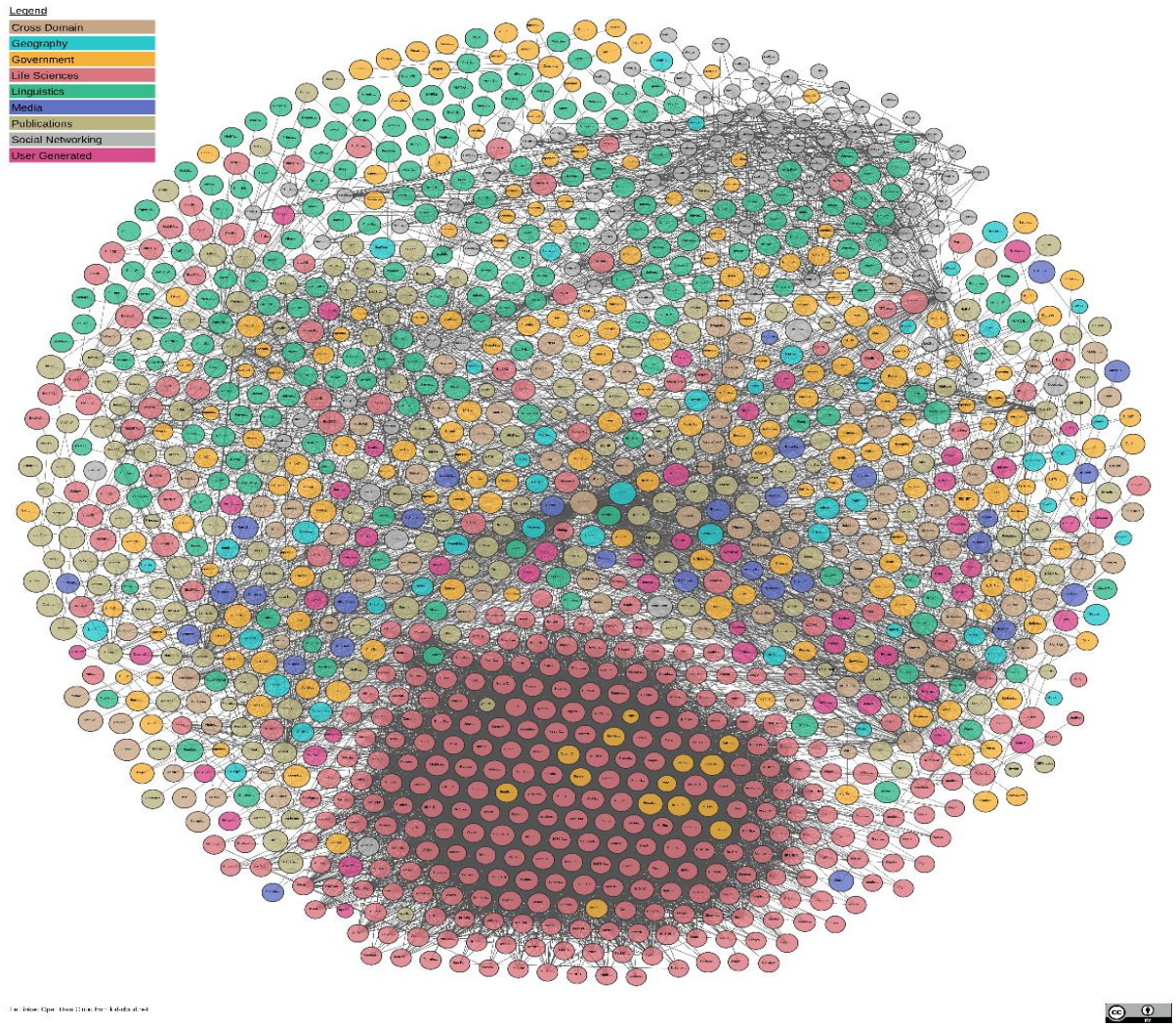

Figure 6: Linked open data cloud: Retrieved from The Linked Open Data Cloud. (2018).

In the LOD cloud there are SPARQL endpoints such as, DBPedia, FactForge, OpenUpLabs, and the European Environment Agency. In addition, the external datasets of LOD can be combined with the reasoning capabilities and spatiotemporal queries for the information retrieval, which is not displayed straight into the BT Hypercat Data Hub (Tachmanzidis, et al, 2017). Describing about Linked Open Data and Internet of Things it is very natural to describe how the Linked Open Data can work with IoT Sensors.

Linked Open Data (LOD) and Sensors

Nowadays there is a huge number of sensors around us, each of them producing a sequence of data items over time for example, a data stream (Phuoc & Hauswirth, 2009). There are sensors inside every mobile phone or any normal personal computer, like, WLAN ,GPS, Bluetooth. There are also numerous sensors working for the environmental domain, for recording the traffic or even logistics

(Phuoc & Hauswirth, 2009). For example, the Hypercat's sensors are used for the environmental domain, collecting data from weather, or traffic and much more. The real data which is taken by sensors and then linked to the Web must be converted into a form of Web resource. When this happens, a URI is assigned for the sensor's data which also allows the construction of links to further Web entities. The same approach can be done to the Stream Feeds, a URL is assigned to every sensor which captures the streaming real-time data or a historical data. This method can also contain the semantic descriptions of every sensor (Phuoc & Hauswirth, 2009). The Figure 7 shows how the URI identifier of each sensor is linked into a virtual RDF graph. The entire graph is not materialized and stored inside a triple storage that's why it is called '*virtual graph*', and it is established by several interlinked subgraphs which are controlled by sensor mashups. In order to describe how the raw data coming from a sensor in a data stream data and linked to a domain knowledge or external linked data the mashup's metadata must be stored into a metadata repository (Phuoc & Hauswirth, 2009). A list of ontologies is used to take the facts and the data models of the sensors. SensorMasher is an approach from (Phuoc & Hauswirth, 2009) following the linked open data principles. This approach can share the data captured by the sensor as a Web data source in order to combined with ease with additional linked-data sources or sensor data.

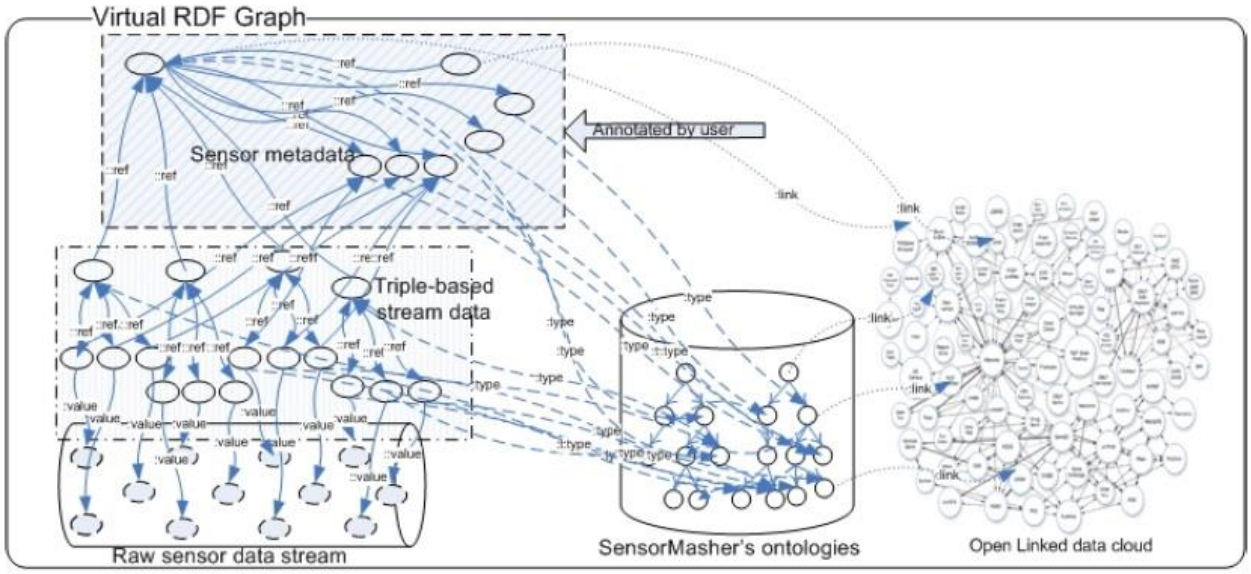

Figure 7:Linking raw data stream to virtual RDF graph: Retrieved from (Phuoc & Hauswirth, 2009)

Linked Open Data (LOD) for Smart Cities

The rapid growth of the population around the world, forced the evolution of cities into Smart Cities. This evolution took the advantage to develop and upgrade the services around the cities, and to improve the way of life into a Smart City. The huge quantity of information accessible helps the better decision-making procedure, evolving the city into a smart and intelligent area to serve its citizens (Murelo et al,

2017). Note that, numerous Open/Closed data sources are accessible through data hubs, this data is a range of structural, statistical and real-time data. This kind of data can be integrated into a unique knowledge base, LOD provides an easier access to the data and supplies a better and more steady expansion of new services and applications. The new Smart City services can retrieve data such as: parking status, traffic status, weather, real-time sensors, transport, different hotels and restaurants, even parks or monuments of the specified city and much more. For example, a Smart city application can help a citizen who is going home by bus, identify the closest supermarket near the bus stops of its bus route. A better example is a service which helps a citizen find the car park which is closer to him and it has empty spaces (Bellini et al, 2014).

Summary

- Semantic web has a huge impact in our lives, transforming the data of the Internet into machine-readable data using its technologies, such as Ontologies, RDF and the JSON-LD.
- JSON-LD is 100% compatible with plain JSON, both serializations are the same syntactically and the only difference is the JSON-LD's keywords.
- There is data that can be retrieved by Sensors using IoT technologies and can be stored into data hubs such as the BT data hub.
- This kind of data can also be added in the LOD cloud to be accessible through several dissimilar sources.
- Using this data, cities can evolve into smart cities, with the help of specific IoT services, for example the usage of the Hypercat specification.
- Therefore, it is very useful to utilize the semantic web description languages to enhance the interoperability of the data into these data hubs and evolve the services such as Hypercat into Semantic services.

Chapter 3

Related Work

There are already some useful tools in the web that translate Semantic web formats, for example RDF/XML to N-Triples or N-triples to JSON-LD. These existing tools are using powerful libraries to parse and serialise syntax and transform the preferred syntax to another. In their work (Stoltz et al, 2013) "RDF Translator: A RESTful Multi-Format Data Converter for the Semantic Web", they discuss of some popular tools including their tool. For example, Any23 is a translator that translate anything to triples using a Java library and a command-line, used by Sindice Semantic Web search engine. Omninator is a HTML service that intends to be a translator for formats having schema.org terms into other syntaxes such as JSON, Turtle or RDF/XML. Nevertheless no one of the above services can support the conversion of traditional RDF formats to RDF embedded inside Web files format. Even in their own project (Stoltz et al, 2013), they discuss of how to parse and serialize RDF syntax to several Semantic web syntaxes such as RDFa, Microdata, RDF/XML, N3, N-Triples, RDF-JSON and JSON-LD. The tool offers the option to upload the file of the syntax and translate it to the preferred syntax or use the website's textbox to paste a part of the syntax and next translate it to the preferred syntax with the help of powerful libraries. They tackle the problem by implementing their converter to work like a Google AppEngine Web service and expand to several aligned requests. The tool uses the RDFLib library to parse, manipulate and serialise RDF graphs. In order to cover a complete conversion cycle, the current RDFLib was lagged of having serializers for Microdata and RDFa. Thus, they tried to use the RDF2RDFa and RDF2Microdata converters, unfortunately the maintenance of these services was stop so their conversions were unsatisfied. In the end they create two plugins well-suited with RDFLib and published them online (Stoltz et al, 2013).

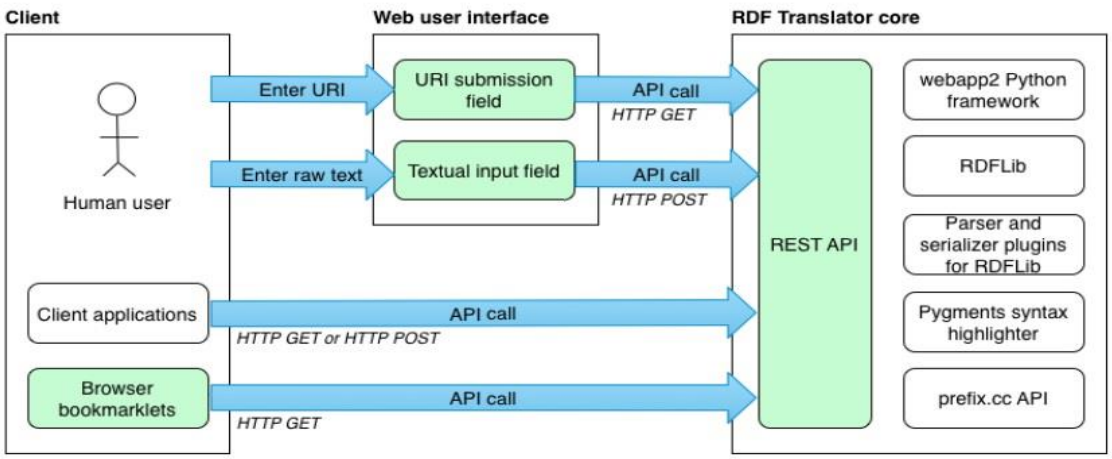

Figure 8: Service Architecture of the RDF Translator: Retrieved from (Stoltz et al, 2013).

In figure 8, there is a description of the architecture of RDF Translator. The REST API is in the centre of the Web service and can handle every received request, such as, directed requests from client applications, browser bookmarklets or any indirect requests through the Web user interface. Both HTTP GET and HTTP POST request methods are supported by the REST API. The Web service can help the users by offering them two very easy to use options, the first one is by providing the URI of the contents which are going to be changed, and a text box input to copy and paste their contents and translate them to their preferred syntax (Stoltz et al, 2013). Although this is a very useful tool to parse and serialize different Semantic web syntaxes, this tool does not cover the translation of plain JSON to JSON-LD. The aforementioned tools do not have the option to parse and serialize JSON syntax to JSON-LD syntax. The problem here is even having a translator from JSON to JSON-LD the translator needs to contain the specific Ontology (in our case the Hypercat Ontology) so it can enhance it to the JSON-LD syntax. Another worth mentioning work is the work of (Tachmanzidis et al, 2016), in their work they translate JSON to RDF(N-Triples) so they give a clear answer of how to work and translate a JSON syntax to another Semantic web syntax. They parse and serialise the plain JSON syntax and by adding the specific Ontology, they transform it to RDF(N-Triples). They use a JavaScript based Web service working with apache Tomcat to load the JSON-based Hypercat calatague and convert it to RDF (N-Triples). The service collects the JSON syntax by the JavaScript function "JSON.parse" and then with call back functions converts the JSON properties to RDF properties adding the ontology and the precise data types. Additionally, there is another function which converts the "in mind" JSON triples to RDF (N-triples), finalizing the conversion. In the end, the final RDF (N-triples) syntax appears in the HTML Web page of the service. Having their point of view, their work of how to collect the JSON syntax and add the Ontology it is very helpful for the conversion of the JSON to JSON-LD syntax. Summarizing these services of how a Semantic web syntax can be converted to a JSON-LD syntax or how a JSON syntax can be converted to a Semantic web syntax shows that this work can be the link between them. However, there is no parser or converter that serializes the transformation of a given Hypercat catalogue from JSON syntax into valid JSON-LD based on a given ontology. What it is needed, is a converter which takes the ontology and convert the JSON-based catalogue into a JSON-LD based one, adding the given ontology and the correct data types to the values. It is also a unique work because it targets the Hypercat Specification and translates the JSON-based Hypercat catalogue to a valid JSON-LD Hypercat catalogue.

Next, in this thesis, there is an explanation of how a clear and valid translation from JSON syntax to JSON-LD syntax can be produced adding the specific Hypercat Ontology, and how the translator works to parse and serialise the JSON syntax to JSON-LD syntax.

Chapter 4

Hypercat JSON to Hypercat JSON-LD

In this chapter, there is a description of the translation of every aspect of Hypercat specification from JSON to JSON-LD. Remember that, JSON-LD is equivalent to JSON and it can support every major RDF concept (Lanthaler & Gült, 2012). The mapping of the JSON properties has already been provided by a previous translation of Hypercat JSON to RDF (Tachmazidis et al, 2016). The Table 1 was used to develop a translator from a JSON-based catalogue to a JSON-LD based catalogue. The core idea behind the translation follows the same rational as for RDF, while the transformation of subjects, predicates and objects follows the JSON-LD format. For instance, for the JSON-based catalogue with the following uri:

"http://portal.bt-hypercat.com/cat"

the *catalogue-metadata*, namely metadata about the catalogue itself:

"rel": "urn:X-hypercat:rels:isContentType"
"val": "application/vnd.hypercat.catalogue+json"

will be translated in JSON-LD, using Table 1:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType": "application/ld+json"

It is worth noting that, the MIME type of the JSON-LD based catalogue is "application/ld+json". The keyword "@id" identifies the unique node objects that are being defined in the file with IRIs (W3C, 2018). Thus, each *Item Object* is translated from "href": "http://api.feed" to "@id": "http://api.feed".

Similar translation should be applied to most of the properties in Table 1, with several exceptions that include *supportsSearch*, which is a property where, the *rel* and *val* are both URIs. Moreover, note that class *Search* represents semantically the numerous types of searches as named individuals, which in JSON-LD should be translated using the keyword "@id".

Therefore, for the JSON-based catalogue "*http://portal.bt-hypercat.com/cat*", the next *catalogue-metadata*:

"rel": "urn:X-hypercat:rels:supportsSearch"
"val": "urn:X-hypercat:search:simple"

will be translated in JSON-LD, using Table 1:

```
"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
 "@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch"}
```

The translation of the property "https://www.w3.org/2000/01/rdf-schema#comment" needs to include both the language of the current document, using the keyword "*@language*", and the text of the description, using the keyword "*@value*". The "*@language*" keyword is to specify the language of the string value which the description is typed, and the "*@value*" keyword is the text of the description (W3C, 2018).

Therefore, for the JSON-based catalogue "*http://portal.bt-hypercat.com/cat*", the next *catalogue-metadata*:

```
"rel": "urn:X-hypercat:rels:hasDescription:en"
"val": "Hypercat DataHub Catalog"
```

will be translated in JSON-LD, using Table 1:

```
"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://www.w3.org/2000/01/rdf-schema#comment" : {
  "@language": "en",
  "@value": "Hypercat DataHub Catalog"}
```

The translation of the property *hypercat:lastUpdated* needs to include both datatypes, using the keyword "*@type*", and relevant data, using the keyword "*@value*".

The "*@type*" keyword is used to set the datatype of a node or typed value. In this situation, the type information is the "*http://www.w3.org/2001/XMLSchema#dateTime*". The "*@value*" keyword is used again to specify the data that is related to the actual property (W3C, 2018).

Thus, the next *Item Object* (in *items*):

```
"href": "http://api.bt-hypercat.com/sensors/feeds/UUID"
```

and "item-metadata" containing:

```
"rel": "urn:X-hypercat:rels:lastUpdated"
```

"val": "2018-05-08T00:00:00Z"

will be translated in JSON-LD, using Table 1:

"@id": "http://api.bt-hypercat.com/sensors/feeds/UUID",
"http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated": {
"@type": "http://www.w3.org/2001/XMLSchema#dateTime",
"@value": "2018-05-08T00:00:00Z"}

The keyword "*@graph*" must be added to the top of the file. The document itself is expanded, which means all prefixes and terms are in full IRIs. The use of this mechanism is very useful since the amount of nodes that exist at the file's top level share the similar context (W3C, 2018). The entire catalogue must be included into a graph's array (*{@graph:[Hypercat JSON-LD catalogue]}*). Note that the keyword "*@context*" is optional, thus it can be omitted within a given catalogue.

For the translation of the BT Data Hub's JSON-based catalogue, the BT Hypercat Ontology has been used as an extension of the core ontology, and is available with the uri:

"http://portal.bt-hypercat.com/ontologies/bt-hypercat"

The namespaces for the *BT Hypercat ontology* can be written with "bt-hypercat:". Thus, the "item-metadata" containing:

"rel": "urn:X-bt:rels:feedTitle",
"val" : "Met Office Datapoint Observations"

this property will be translated to JSON-LD using the *BT Hypercat ontology* and not the core Hypercat ontology:

"@id": "http://api.bt-hypercat.com/sensors/feeds/UUDI",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_title":
"Met Office Datapoint Observations"

One more change which must take place from the JSON-based BT Hypercat to JSON-LD-based is the type class:

"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"

The above class shows the type of each feed, so for the JSON-LD must change to the keyword “@type”. Again, for the item-metadata containing :

```
"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
"val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#sensors"
```

will be translated to the following:

```
"@id": "http://api.bt-hypercat.com/sensors/feeds/UUDI",
"@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#sensors"
```

Duplicate keys are not allowed in JSON-LD. In order to avoid such duplicates in any given JSON-LD based catalogue, catalogue's and items' descriptions must be stored as a separate JSON object (one per unique subject), while each property must be included once with multiple values being added into an array.

For example, if we get the following *supportsSearch*:

```
{"rel": "urn:X-hypercat:rels:supportsSearch",
  "val": "urn:X-hypercat:search:simple" },
{"rel": "urn:X-hypercat:rels:supportsSearch",
  "val": "urn:X-hypercat:search:geobound" }
```

will be translated in JSON-LD: (storing supported searches into an array): the searches must be entered into an array and the property must be provided once, like the following:

```
  "@id": "http://portal.bt-hypercat.com/cat-json-ld",
  "http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": [
    {"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch"},
    {"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#GeoboundSearch"}]
```

the same goes for every property that is repeated in a given *Catalogue* such as hypercat:isContentType:

```
{"rel":  "urn:X-hypercat:rels:isContentType",
  "val": "application/json"},
{"rel":  "urn:X-hypercat:rels:isContentType",
  "val": "application/xml"},
```

```
{"rel":  "urn:X-hypercat:rels:isContentType",
  "val": "text/xml"},
{"rel":  "urn:X-hypercat:rels:isContentType",
  "val": "text/csv"}
```

and the translation is the following:

```
"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType": [
        "application/json",
        "application/xml",
        "text/xml",
        "text/csv"
      ]
```

For properties that are not provided in Table 1, a Hypercat JSON-LD developer is encouraged to create his own OWL ontology by extending the ontology defined by (Tachmazidis et al, 2016) in this paper: "*Hypercat RDF: Semantic Enrichment for IoT"*. Therefore, the brand-new ontology has the ability to catch the concept of a developer's catalogue, by supplying new properties, and would make possible for the complete conversion of the developer's current JSON-based catalogue into a JSON-LD based catalogue. Otherwise, Hypercat JSON-LD developers could develop and translate only the main properties, using Table 1. Although in this occasion, the JSON-LD based catalogue would have less data in contrast to the JSON-based catalogue, it would not stop being a minimal yet valid Hypercat JSON-LD catalogue.

Hypercat BT JSON-based to JSON-LD based

The following examples are the JSON-based properties from the core Hypercat and how with the help of Table 1, a developer can translate them from JSON property to RDF property, and then use them for the JSON-LD translation with the full ontology uri:

Table 1.

1) "rel":"urn:X-hypercat:rels:hasDescription:en",
"val":"Test Hypercat Catalogue"
to :

"https://www.w3.org/2000/01/rdf-schema#comment ": {
"@language": "en",
"@value": "Test Hypercat Catalogue"}

2) { "rel":"urn:X-hypercat:rels:isContentType",
"val":"application/vnd.hypercat.catalogue+json"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType":
"application/vnd.hypercat.catalogue+json"

3) {"rel":"urn:X-hypercat:rels:hasHomepage",
"val":"http://home.page.com"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#hasHomepage": {
"@id": "http://home.page.com"}

4) {"rel":"urn:X-hypercat:rels:containsContentType",
"val":"text/xml"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#containsContentType": "text/xml"

5) {"rel": "urn:X-hypercat:rels:hasLicense",
"val": "http://licence.html"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#hasLicense": {
"@id": "http://licence.html"}

6) {"rel": "urn:X-hypercat:rels:acquireCredential",
"val": "http://request.credentials.html"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#acquireCredential": {
"@id": "http://request.credentials.html"}

7) {"rel": "urn:X-hypercat:rels:eventSource",
"val": "http://event.source.html"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#eventSource": {
"@id": "http://event.source.html"}

8) { "rel": "urn:X-hypercat:rels:hasRobotstxt",
"val": "http://catalogue/robots.txt"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#hasRobotstxt": {
"@id": "http://catalogue/robots.txt"}

9) {"rel": "urn:X-hypercat:rels:accessHint",
"val": "http://access.hint.html"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#accessHint": {
"@id": "http://access.hint.html"}

10) {"rel": "urn:X-hypercat:rels:lastUpdated",
"val": "2017-12-01T00:00:00Z"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated": {
"@type": "xsd:dateTime",
"@value": "2017-12-01T00:00:00+00:00"}

The following are the search individuals of the catalogue. To avoid the duplicate key found the search individuals must be entered into an array with a single key (predicate).

11) {"rel":"urn:X-hypercat:rels:supportsSearch",
"val":"urn:X-hypercat:search:simple"},

12) {"rel":"urn:X-hypercat:rels:supportsSearch",
"val":"urn:X-hypercat:search:geobound"},

13) {"rel":"urn:X-hypercat:rels:supportsSearch",
"val":"urn:X-hypercat:search:lexrange"},

14) {"rel":"urn:X-hypercat:rels:supportsSearch",
"val":"urn:X-hypercat:search:multi"},

15) {"rel":"urn:X-hypercat:rels:supportsSearch",
"val":"urn:X-hypercat:search:prefix"},

16) {"rel":"urn:X-hypercat:rels:supportsSearch",
"val":"urn:X-hypercat:search:semantic"}

to:

"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": [
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#GeoboundSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#LexrangeSearch"},
{"@id": http://portal.bt-hypercat.com/ontologies/hypercat#MultiSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#PrefixSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SemanticSearch"}]

Next, we have all BT Hypercat JSON-based catalogue properties where we use again the ontology from the RDF translation, to translate them to JSON-LD. The following are template examples with every possible translation from BT Hypercat JSON-based catalogue to BT Hypercat JSON-LD-based catalogue.

For the first template the property is defined in a full URI and an object that is a string value:

Example : JSON to JSON-LD.

JSON Property: {"rel": "urn:X-hypercat:rels:isContentType",
"val": "application/vnd.hypercat.catalogue+json"}

JSON-LD Property: "http://portal.bt-hypercat.com/ontologies/hypercat#isContentType":
"application/ld+json"

Core Hypercat

1. "http://portal.bt-hypercat.com/ontologies/hypercat#isContentType": "application/ld+json",
2. "http://portal.bt-hypercat.com/ontologies/hypercat#containsContentType": "text/xml",

Class: Feed

1. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_id": "feedId",
2. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_creator": "feedCreator",
3. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_title": "feedTitle",
4. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_status": "feedStatus",
5. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_private": "feedPrivate",
6. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_description": "feedDescription",
7. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_email" : "feedEmail",
8. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_tag": "feedTag",

9. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_location_name": "feedLocationName",
10. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_exposure": "feedExposure",
11. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_domain" : "feedDomain",
12. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_disposition": "feedDisposition",
13. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_the_geom": "feedTheGeom",

Class: DataStream

1. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_id": "datastreamId",
2. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "datastreamTag",
3. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_max_value": "datastreamMaxValue",
4. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_min_value": "datastreamMinValue",
5. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol": "datastreamUnitSymbol",
6. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "datastreamUnitText",

The second template is where the object is a string/dateTime, so the "@type" keyword must be added with the "http://www.w3.org/2001/XMLSchema#dateTime" to define the dateTime type:

Example : JSON to JSON-LD.

JSON Property: {"rel": "urn:X-hypercat:rels:lastUpdated",
"val": "2018-05-08T00:00:00Z"}

JSON-LD Property: "http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated": {
"@type": "http://www.w3.org/2001/XMLSchema#dateTime",
"@value": "2018-05-08T00:00:00Z"}

Core Hypercat

- "http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated": {
"@type":"http://www.w3.org/2001/XMLSchema#dateTime",
"@value": "2018-05-08T00:00:00Z"},

Class: Feed

- "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_updated": {
  "@type":"http://www.w3.org/2001/XMLSchema#dateTime",
  "@value": "2018-05-08T00:00:00Z"},

Class: DataStream

- "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_current_time": {
  "@type":"http://www.w3.org/2001/XMLSchema#dateTime",
  "@value": "datastreamCurrentTime"},

The third template is where that is necessary to add the object as a string/anyURI , so the "@type" keyword must be added with the "http://www.w3.org/2001/XMLSchema#anyURI" to define the anyURI type.

Example : JSON to JSON-LD.

JSON Property: {"rel":"urn:X-bt:rels:feedUrl",
"val":"/feed/uri"}

JSON-LD Property: "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_url": {
"@type":"http://www.w3.org/2001/XMLSchema#anyURI",
"@value": "/feed/uri"}

Class: Feed

1. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_url": {
   "@type":"http://www.w3.org/2001/XMLSchema#anyURI",
   "@value": "/feed/uri"},
2. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_icon": {
   "@type":"http://www.w3.org/2001/XMLSchema#anyURI",
   "@value": "/feed/icon"},
3. "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_website": {
   "@type":"http://www.w3.org/2001/XMLSchema#anyURI",
   "@value": "feed/website"},

The following template is the one where the object is a URL and must come with the keyword "@id" and the following are the properties where their values should be translated with this template:

Example : JSON to JSON-LD.

JSON Property: {"rel": "urn:X-hypercat:rels:hasLicense",
"val": "http://licence.html"}

JSON-LD Property: "http://portal.bt-hypercat.com/ontologies/hypercat#hasLicense": {
"@id": "http://licence.html"}

Core Hypercat

1. "http://portal.bt-hypercat.com/ontologies/hypercat#hasHomepage": {
   "@id": "http://home.page.com"},
2. "http://portal.bt-hypercat.com/ontologies/hypercat#hasLicense": {
   "@id": "http://licence.html"},
3. "http://portal.bt-hypercat.com/ontologies/hypercat#acquireCredential": {
   "@id": "http://request.credentials.html"},
4. "http://portal.bt-hypercat.com/ontologies/hypercat#eventSource": {
   "@id": "http://event.source.html"},
5. "http://portal.bt-hypercat.com/ontologies/hypercat#hasRobotstxt": {
   "@id": "http://catalogue/robots.txt"},
6. "http://portal.bt-hypercat.com/ontologies/hypercat#accessHint": {
   "@id": "http://access.hint.html"},

Class: EventStream

- "http://portal.bt-hypercat.com/ontologies/bt-hypercat#hasEventStream": {
  "@id": "http://api.test.datastream"},

Class: EventFeed

- "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToEventFeed": {
  "@id": "http://api.test.feed"},

Class: SensorStream

- "http://portal.bt-hypercat.com/ontologies/bt-hypercat#hasSensorStream": {
  "@id": "http://api.test.datastream"},

Class: SensorFeed

- "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed": {
  "@id": "http://api.test.feed"},

The “@id” template must also be used for the supportSearch property, but if there are more than one search types then they must be entered in an array just like the following example:

"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": [
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#GeoboundSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#LexrangeSearch"},
{"@id": http://portal.bt-hypercat.com/ontologies/hypercat#MultiSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#PrefixSearch"},
{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SemanticSearch"}]

The last template is for the property of types of each class and comes with the keyword "@type" this comes for each JSON-based (href) where in the JSON-LD changes to "@id" and it defined the type of each "@id":

Example : JSON to JSON-LD.

JSON Property: {"rel":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
"val":"http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventFeed"}
JSON-LD Property: "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventFeed"

Class: EventFeed

For every "EventFeed" should have the following JSON-LD @type:

"rel":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
"val":"http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventFeed"

to: "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventFeed"

Class: LocationFeed

Every "LocationFeed" should have the following JSON-LD @type:

"rel":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
"val":"http://portal.bt-hypercat.com/ontologies/bt-hypercat#LocationFeed"

to: "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#LocationFeed"

Class: SensorFeed

Every "SensorFeed" should have the following JSON-LD @type:

"rel":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
"val":"http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed"

to: "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed"

Class: EventStream

Every "EventStream" should have the following JSON-LD @type:

"rel":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
"val":"http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventStream"

to: "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventStream"

Class: SensorStream

Every "SensorStream" should have the following JSON-LD @type:

"rel":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
"val":"http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"

to: "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"

Example from JSON to JSON-LD based catalogue

This is an example from a BT Hypercat JSON based catalogue to a BT Hypercat JSON-LD based catalogue to help the readers of the above translation understand how it works. It starts with the introduction of a BT Hypercat JSON-based catalogue sample, with several search types into the catalogue following by items. The full catalogue can be found here: "https://portal.bt-hypercat.com/cat". In the example there are some explanatory comments to help the readers.

{ "catalogue-metadata":[
{"rel": "urn:X-hypercat:rels:isContentType",
"val": "application/vnd.hypercat.catalogue+json"},
{"rel": "urn:X-hypercat:rels:hasDescription:en",
"val": "BT Hypercat DataHub Catalog"},
{"rel": "urn:X-hypercat:rels:supportsSearch",
"val": "urn:X-hypercat:search:simple"},
{"rel": "urn:X-hypercat:rels:supportsSearch",
"val": "urn:X-hypercat:search:prefix"},
{"rel": "urn:X-hypercat:rels:supportsSearch",
"val": "urn:X-hypercat:search:multi"},
{"rel": "urn:X-hypercat:rels:supportsSearch",
"val": "urn:X-hypercat:search:lexrange"},
{"rel": "urn:X-hypercat:rels:supportsSearch",
"val": "urn:X-hypercat:search:geobound"}],

(In the catalogue-metadata it can be seen the document's type at the top, followed by the various search types of the catalogue. The items follow with their unique "href" and each item has its "item-metadata" (The Items's Information). The item-metadata describes the data of each "href" and it seems like a small list. <u>Each item will have different colour to help the reader identify them into this example.</u>)

"items":[
{"href": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2",
"item-metadata":[{
"rel": "urn:X-hypercat:rels:hasDescription:en",
"val": "Met Office Datapoint UK hourly site-specific observations for Killowen as recorded in real time by the Met Office UK Monitoring System. Parameters reported are based on the instrumentation installed at each site."},
{"rel": "urn:X-bt:rels:feedTitle",
"val": "Met Office Datapoint Observations - 99015 (Killowen)"},
{"rel": "urn:X-bt:rels:feedTag",
"val":"weather"},
{"rel": "urn:X-bt:rels:hasSensorStream",
"val":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0"},
{"rel": "urn:X-bt:rels:hasSensorStream",

"val":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1"},

{"rel": "urn:X-bt:rels:hasSensorStream",

"val":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2"},

{"rel": "urn:X-bt:rels:hasSensorStream",

"val":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3"},

{"rel": "http://www.w3.org/2003/01/geo/wgs84_pos#lat",

"val":"54.067"},

{"rel": "http://www.w3.org/2003/01/geo/wgs84_pos#long",

"val":"-6.183"},

{"rel": "urn:X-hypercat:rels:isContentType",

"val": "application/json"},

{"rel": "urn:X-hypercat:rels:isContentType",

"val": "application/xml"},

{"rel": "urn:X-hypercat:rels:isContentType",

"val": "text/xml"},

{"rel": "urn:X-hypercat:rels:isContentType",

"val": "text/csv"},

{"rel": "urn:X-hypercat:rels:hasLicense",

"val": "http://reference.data.gov.uk/id/open-government-licence"},

{"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",

"val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed"},

{"rel": "urn:X-hypercat:rels:lastUpdated",

"val": "2017-02-22T17:36:32Z"}

]},

(The 'href' bellow (Green) is an item coming from the item above, this 'href' is a datastream which belongs to the above item (Red). Thus, each datastream again will have its own metadata. Every 'href' below will have its metadata described like a list and all of them are part of the first item.)

{"href": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0",

"item-metadata":[{

"rel": "urn:X-bt:rels:belongsToSensorFeed",

"val": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},

{"rel": "urn:X-bt:rels:datastreamMaxValue",

```
      "val": "max_value=50"},
    {"rel": "urn:X-bt:rels:datastreamMinValue",
      "val": "min_value=-30"},
    {"rel": "urn:X-bt:rels:datastreamUnitText",
     "val": "degrees Celsius"},
    {"rel": "urn:X-bt:rels:datastreamUnitSymbol",
     "val":"degrees Celsius"},
    {"rel": "urn:X-bt:rels:datastreamUnitType",
       "val":"derivedSI"},
    {"rel": "urn:X-bt:rels:datastreamTag",
      "val":"temperature"},
    {"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
      "val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
  ]},
{"href": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1",
 "item-metadata":[{
       "rel": "urn:X-bt:rels:belongsToSensorFeed",
     "val": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
    {"rel": "urn:X-bt:rels:datastreamUnitText",
     "val": "16 point compass"},
    {"rel": "urn:X-bt:rels:datastreamUnitType",
      "val": "contextDependentUnits"},
    {"rel": "urn:X-bt:rels:datastreamTag",
     "val": "winddirection"},
    {"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
     "val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
  ]},
  {"href": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2",
   "item-metadata":[{
       "rel": "urn:X-bt:rels:belongsToSensorFeed",
       "val": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
    {"rel": "urn:X-bt:rels:datastreamUnitText",
      "val": "miles per hour"},
    {"rel": "urn:X-bt:rels:datastreamUnitSymbol",
```

```
      "val":"mph"},
    {"rel": "urn:X-bt:rels:datastreamUnitType",
     "val":"derivedUnits"},
    {"rel": "urn:X-bt:rels:datastreamTag",
     "val":"windspeed"},
    {"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
     "val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
  ]},
{"href": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3",
  "item-metadata":[{
       "rel": "urn:X-bt:rels:belongsToSensorFeed",
       "val": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
    {"rel": "urn:X-bt:rels:datastreamUnitText",
     "val": "miles per hour"},
    {"rel": "urn:X-bt:rels:datastreamUnitSymbol",
     "val": "mph"},
    {"rel": "urn:X-bt:rels:datastreamUnitType",
     "val": "derivedUnits"},
   {"rel": "urn:X-bt:rels:datastreamTag",
     "val":"windspeed"},
    {"rel": "urn:X-bt:rels:datastreamTag",
     "val":"gust"},
    {"rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
     "val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
  ]}
 ]}
```


Furthermore, it continues with the same Hypercat catalogue sample but in JSON-LD based translation. It can be seen that the whole catalogue now is inside a graph, the MIME type of the catalogue has changed to JSON-LD and there are also changes to the properties showing their full URI by following the ontology's changes of namespaces. In addition, a new property *bt-hypercat:hasItem* took place into the beginning of the catalogue having all items in an array (@id/ subject, ID's). This is happening to serve the ontology's property "*hasItem*".

{"@graph" : [{

"@id": "http://localhost:8080/Hypercat-jsonld-full1/cat/Test1.json",

"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType" : "application/ld+json",

"https://www.w3.org/2000/01/rdf-schema#comment": {"@language" : "en",

"@value": "BT Hypercat DataHub Catalog"},

"http://portal.bt-hypercat.com/ontologies/hypercat#hasItem": [

{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0"},

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1"},

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2"},

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3"}

],

"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch" : [

{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch"},

{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#PrefixSearch"},

{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#MultiSearch"},

{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#LexrangeSearch"},

{"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#GeoboundSearch"}]

},

(Above, it can be seen the catalogue's metadata translation. Everything now looks more structured. First there is a small description of the document with the name and type. Then there is a *hasItem* property with a list of the items and then there is the *supportSearch* property with the list of the various search types. The items follow each item-metadata and are inside the same curly-bracket so the 'href' now is translated into '@id' and the metadata are following in the same curly-bracket. Each item will have different colour to help the reader identify them into this example.)

{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2",

"https://www.w3.org/2000/01/rdf-schema#comment" : {"@language" : "en",

"@value": "Met Office Datapoint UK hourly site-specific observations for Killowen as recorded in real time by the Met Office UK Monitoring System. Parameters reported are based on the instrumentation installed at each site."},

"http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_title":"Met Office Datapoint Observations - 99015 (Killowen)",

"http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_tag" : "weather",

```
    "http://portal.bt-hypercat.com/ontologies/bt-hypercat#hasSensorStream" :  [
  {"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-
0076292fd4a2/datastreams/0"},
  {"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-
0076292fd4a2/datastreams/1"},
  {"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-
0076292fd4a2/datastreams/2"},
  {"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3"}
 ],
   "http://www.w3.org/2003/01/geo/wgs84_pos#lat" :   "54.067",
    "http://www.w3.org/2003/01/geo/wgs84_pos#long" :   "-6.183",
"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType" :  [
           "application/json",
          "application/xml",
          "text/xml",
           "text/csv"],
    "http://portal.bt-hypercat.com/ontologies/hypercat#hasLicense" :   {
     "@id": "http://reference.data.gov.uk/id/open-government-licence"},
     "@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed",
     "http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated" :   {
     "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
     "@value": "2017-02-22T17:36:32Z"}
  },
   {"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0",
     "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :   {
     "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_max_value":"max_value=50",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_min_value":"min_value=-30",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "degrees Celsius",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol":"degrees Celsius",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type":  "derivedSI",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "temperature",
      "@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"},
  {"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1",
     "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :   {
```

```
        "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2" },
       "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text":"16point compass",
       "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type":   "contextDependentUnits",
       "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "winddirection",
       "@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"},
    {"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2",
       "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :   {
        "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "miles per hour",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol": "mph",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type": "derivedUnits",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "windspeed",
      "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"},
    {"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :   {
       "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "miles per hour",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol": "mph",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type": "derivedUnits",
      "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag" :  [
           "windspeed",
           "gust"],
       "@type" :  "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
  ]
}
```


Thus, following the above example of the BT Hypercat JSON to JSON-LD based catalogue and the ontology, anyone can translate its catalogue easily without any problems.

Hypercat JSON-based to JSON-LD based catalogue parser

For the complete translation of a given Hypercat JSON-based catalogue to JSON-LD based catalogue two parsers have been developed using the JavaScript programming language. The parsers are running through the Apache Tomcat server. The parsers collect the text from the JSON file where the catalogue is inside, then with the help of specific functions, it sets (in mind) the *href*, the *properties* and the

objects of the JSON-based catalogue to a triple-centric text. For example, the JSON-based Hypercat will look like the following in the JSON file:

```
{"href": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1",
  "item-metadata":[{
      "rel": "urn:X-bt:rels:belongsToSensorFeed",
      "val": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"
    },{
      "rel": "urn:X-bt:rels:datastreamUnitText",
      "val": "16 point compass"
    },{
      "rel": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
      "val": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
]}
```

The triple-centric will look like the following, the first line is the subject following by the predicate and in the end is the object:

```
(SUBJECT)http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1
(PREDICATE) urn:X-bt:rels:belongsToSensorFeed
(OBJECT) http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2

(SUBJECT)http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1
(PREDICATE)urn:X-bt:rels:datastreamUnitText
(OBJECT) 16 point compass

(SUBJECT)http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1
(PREDICATE) http://www.w3.org/1999/02/22-rdf-syntax-ns#type
(OBJECT) http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream
```

Additionally, the JSON-LD text can be created by JavaScript tables, so each table can have a unique subject/href following with the predicates/rels and objects/vals of the specified subject/predicate just like the JSON-LD. The following example it is how the (in mind) process of the parser works. It can be seen that in the first row there is the Subject-1 following by Predicate-1 and Object-1, but in the second row the Subject-1 is deleted. This is because is the same subject, so it does not need to be

shown in the JSON-LD syntax again. Then the Predicate-2 and Object 2 follows. The same goes for the predicates. If a predicate is the same for the same subject just like the Subject-2, then the parser deletes the predicate and shows only the object. To be more precise, the Object-2 and Object-3 of Subject-2 will be set into an array to avoid the 'duplicate key' of JSON-LD.

SUBJECT	**PREDICATE**	**OBJECT**
SUBJECT-1	PREDICATE-1	OBJECT-1
~~**SUBJECT-1**~~	PREDICATE-2	OBJECT-2
~~**SUBJECT-1**~~	PREDICATE-3	OBJECT-3
SUBJECT-2	PREDICATE-1	OBJECT-1
~~**SUBJECT-2**~~	PREDICATE-2	OBJECT-2
~~**SUBJECT-2**~~	~~PREDICATE-2~~	OBJECT-3
~~**SUBJECT-2**~~	PREDICATE-3	OBJECT-4

Meanwhile, it transforms the plain properties to full URIs following the specified ontology. This triple-centric text is now the *subjects*, *predicates* and *objects* of the JSON-LD based catalogue, but without the correct keywords, it reminds of TURTLE. Then with another function, the parser sets the specific JSON-LD keywords such as *"@id"*, *"@type"* and *"@value"* to complete a valid translation. In the end, the parser prints the JSON-LD based catalogue to the HTML page. The following is an example of JSON-LD based:

```
{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1",
"@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed": {
   "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2" },
 "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text":"16 point compass"}
```

The parsers also collect the unique subjects of every item into the catalogue to add them into an array to the beginning of the catalogue to serve the ontology's property *hasItem.* Thus, every catalogue will have a *hasItem* property in the beginning just like the following example:

```
"http://portal.bt-hypercat.com/ontologies/hypercat#hasItem": [
{"@id": "http://api.bt-hypercat.com/events/feeds/ee9d9cd9-6953-4584-a59f-ea6fe5156ebb"},
```

{"@id": "http://api.bt-hypercat.com/events/feeds/ee9d9cd9-6953-4584-a59f-ea6fe5156ebb/datastreams/1"},
{"@id": "http://api.bt-hypercat.com/events/feeds/ee9d9cd9-6953-4584-a59f-ea6fe5156ebb/datastreams/2"},
{"@id": "http://api.bt-hypercat.com/events/feeds/ee9d9cd9-6953-4584-a59f-ea6fe5156ebb/datastreams/3"},
{"@id": "http://api.bt-hypercat.com/events/feeds/ee9d9cd9-6953-4584-a59f-ea6fe5156ebb/datastreams/4"},
{"@id": "http://api.bt-hypercat.com/sensors/feeds/a78d4c54-15ec-4a60-a7e8-492fc36143d0"}]

The second parser does the same job with the first one but with slight differences. It can translate every JSON-based catalogue to 'extended' JSON-LD based catalogues, this means that every object it is into an array for the developers who want the fully expanded form of their Hypercat catalogues.

{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1",
"@type": ["http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"],
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed": [{ "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2" }],
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": ["16 point compass"]}

A future function that will be added to the parsers is to download the catalogue to a JSON-LD file. The parsers have been also uploaded to the Github's website to encourage the large community of developers to use them and translate their Hypercat JSON-based catalogues to JSON-LD based. This is the link to the Github repository which includes the parsers "https://github.com/Mikegeo/hypercat-json-ld-parser".

Chapter 5

Hypercat JSON-LD Specification

In this chapter, there is the inspection of every feature of the Hypercat JSON-LD specification by following the structures and the examples of the (Tachmazidis et all, 2016) and (Beart, 2016). Most of them might be same because there were not many differences between the search types or parameters and their meaning.

Hypercat File Format Specification: In the previous chapter, the details of how to create or translate a JSON-LD based catalogue, with all core semantic properties and their equivalent to JSON properties have been presented. Nevertheless, additional aspects need to be covered as well. For example, every instance of class *MetadataAnnotator* requires the mandatory property "https://www.w3.org/2000/01/rdf-schema#comment" and might contain the non-compulsory properties such as "hypercat:containsContentType", "hypercat:hasHomepage", "hypercat:isContentType" and "hypercat:supportsSearch". Moreover, the compulsory property "hypercat:isContentType" must be included in all instances of the Catalogue. However, according to the previous chapter, each JSON-LD based catalogue has been defined in JSON-LD format with the MIME type to “application/ld+json”. The same terms of extensibility are followed along the lines of the Hypercat 3.00 specification: (a) when a metadata relationship is unidentified should be ignored, (b) new search type can be added but only when a catalogue server supports it, (c) a different language, for human readable descriptions can be added, (d) old version catalogues can point to a new version, and the opposite, without type ambiguity, and (e) the catalogue can also have any amount of other properties and classes, if the developers want to fit in, but firstly they need to define them into an ontology (Tachmazidis et all, 2016).

Hypercat Server API Specification: Any server who supports a JSON-LD based Hypercat should follow the JSON-based Hypercat server specification with slight differences. Any JSON-LD based catalogue server should deliver a “/cat-json-ld” endpoint which is easily readable to the public, and serves a Hypercat file declared in JSON-LD. The features such as update, insert and delete in the JSON-LD based Hypercat server, can be executed in the similar method as for a JSON-based Hypercat server, but the result will be a JSON-LD based catalogue, not a JSON-based catalogue. The search mechanisms, such as simple or multi can be performed in the same technique, on the top of the JSON-LD based catalogues. In a JSON-LD based catalogue with the given url of "*http://portal.bt-hypercat.com/cat-json-ld*", the simple search mechanism can be included like the following:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch"}

Every query parameter must be in URL form and each of them are optional. For several search parameters, the server must return the crossing of items where the search parameters match in an item, merging parameters with boolean AND (Tachmazidis et all, 2016). The Simple search provides the next parameters:

i. "*s*", which represents the JSON-LD subject into the linked data graph.
ii. "*p*", which represents the property (predicate in RDF).
iii. "*o*", which represents the JSON-LD object or value.

Take notice that, if the (c) "*o*", is labelled by an IRI, then it is called object, but if it is labelled by something else that is not IRI such as numbers, it is called value (Lanthaler & Gült, 2012). The following is an example of how we can query a given catalogue. Note that every query parameter must be in URL form and each of them are optional.

The following query:
?s= http://api.bt-hyperat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2
will return any item matching the search criteria:


```
{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2",
  "http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :  {
   "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
 "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "miles per hour",
 "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol": "mph",
 "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type": "derivedUnits",
 "http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "windspeed",
 "@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
```

The following query:

?p=http://portal.bt-hypercat.com/ontologies/ hypercat#hasLicense

&o= http://reference.data.gov.uk/id/open-government-licence

will return any item matching the search criteria:

{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2",

"https://www.w3.org/2000/01/rdf-schema#comment" : {"@language" : "en",

"@value": "Met Office Datapoint UK hourly site-specific observations for Killowen as recorded in real time by the Met Office UK Monitoring System. Parameters reported are based on the instrumentation installed at each site."},

"http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_title":"Met Office Datapoint Observations - 99015 (Killowen)",

"http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_tag" : "weather",

"http://portal.bt-hypercat.com/ontologies/bt-hypercat#hasSensorStream" : [

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0"},

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1"},

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2"},

{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3"}

],

"http://www.w3.org/2003/01/geo/wgs84_pos#lat" : "54.067",

"http://www.w3.org/2003/01/geo/wgs84_pos#long" : "-6.183",

"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType" : [

"application/json",

"application/xml",

"text/xml",

"text/csv"],

"http://portal.bt-hypercat.com/ontologies/hypercat#hasLicense" : {

"@id": "http://reference.data.gov.uk/id/open-government-licence"},

"@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed",

"http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated" : {

"@type": "http://www.w3.org/2001/XMLSchema#dateTime",

"@value": "2017-02-22T17:36:32Z"}

}

The following query:

?o=2017-02-22T17:36:32Z

will return any item matching the search criteria:


```
{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2",
    "https://www.w3.org/2000/01/rdf-schema#comment" :  {"@language" : "en",
     "@value": "Met Office Datapoint UK hourly site-specific observations for Killowen as recorded in real time by the Met Office UK Monitoring System. Parameters reported are based on the instrumentation installed at each site."},
    "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_title":"Met Office Datapoint Observations - 99015 (Killowen)",
    "http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_tag" :  "weather",
    "http://portal.bt-hypercat.com/ontologies/bt-hypercat#hasSensorStream" : [
  {"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0"},
  {"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1"},
  {"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2"},
  {"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3"}
 ],
   "http://www.w3.org/2003/01/geo/wgs84_pos#lat" :  "54.067",
    "http://www.w3.org/2003/01/geo/wgs84_pos#long" :  "-6.183",
"http://portal.bt-hypercat.com/ontologies/hypercat#isContentType" : [
           "application/json",
          "application/xml",
          "text/xml",
           "text/csv"],
    "http://portal.bt-hypercat.com/ontologies/hypercat#hasLicense" :  {
     "@id": "http://reference.data.gov.uk/id/open-government-licence"},
     "@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed",
     "http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated" :  {
     "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
     "@value": "2017-02-22T17:36:32Z"}
  }
```

The following query:

?p=@type&o=http://portal.bt-hypercat.com/ontologies/bt-hypercat#Datastream

will not return any item since no item is of type *Datastream*. Note that simple search cannot extract information from the underlying ontology, thus ignoring items of type *SensorStream* and *EventStream* (classes *SensorStream* and *EventStream* are subclasses of *Datastream*).

Hypercat Subscription: The specification of "Hypercat 3.00" defines a straightforward subscription service, delivering an API for polling catalogues. The client with the subscription of a Hypercat server which provides a stream of events, will obtain a stream of events and every event will contain an event body and name. Firstly, by fetching a catalogue and then accumulate the events that are in the catalogue, a client can keep a synchronized version of the given catalogue. A given JSON-LD based catalogue can use the same Hypercat subscription mechanism with the JSON-based but with slight changes. A JSON-LD based catalogue that can be use subscription, must have the property "*hypercat:eventsource*". For events of a specific catalogue item in the catalogue, the event name is the (unique) JSON-LD's subject (coming with the keyword @id) for the stipulated item. Furthermore, the event body for an item update event is the nest of JSON-LD properties that are associated to the specified item, whereas, the event body is a blank string for an item deletion event.

Hypercat Resource Subscription: The link to resources over URIs is one of the capabilities of Hypercat. This kind of resources might include real-time data compulsory for numerous applications. In the field of IoT, there are numerous use-cases where client applications required real-time data feeds from devices (sensors), hubs (Datahubs) and further services. A previous option was to place all data directly into Hypercat catalogues, but the idea was unsuitable because of the plain data model of a JSON-based catalogue. In contrast, a JSON-LD based catalogue could solve the problem of inserting data straight inside the Hypercat catalogue, assuming that the entered data is semantically enhanced and is provided in JSON-LD syntax.

Hypercat Signing: Hypercat Signing for a JSON-LD based catalogue can be done in an analogous way to a JSON-based catalogue, since JSON-LD is fully compatible with plain JSON (Lanthaler & Gült, 2012). For each item in the JSON-LD based catalogue, the signature may be added after the unique item's @id. In this case, the digest can cover the @id and the item. For the entire Hypercat JSON-LD based catalogue, the signature may be added to the top of the document under the catalogue's @id. In this case, the digest covers both catalogue and all items. A detailed description of signing and verification is referred to future work.

Hypercat Security Access Hints: The systems that support Hypercat must deliver open data with crossed links when is achievable. However, there are numerous systems that will possibly deliver data only when the client is authenticated. Furthermore, where data such as resources or catalogues are

discoverable, but without the authentication the data are not accessible. The authentication information can be presented to the clients by the property "*hypercat:accessHint*". This property should point to either machine or human readable description. A related point to consider is that, if there are multiple "*hypercat:accessHint*" statements, the client must presume that the data can be retrieved using multiple verification systems.

Hypercat Security Credential Acquisition: To have the opportunity to use data such as, catalogues and resources, a Hypercat catalogue can support numerous methods of acquiring access credentials. Consequently, a catalogue or item, can use the property "*hypercat:acquireCredential*" to point a webpage which can describe itself or a resource that will help the client to acquire credentials. A related point to consider is that if there are multiple "*hypercat:acquireCredential*" declarations, the client must presume that credentials can be acquire-able in several ways.

Hypercat Geographic Bounding Box Search: The geographic bounding box search gives the permission to filter the items that fall inside a geographic area, which can be defined by a bounding box. The JSON-LD properties that a JSON-LD based catalogue can use again are:

"http://www.w3.org/2003/01/geo/wgs84pos#lat"
"http://www.w3.org/2003/01/geo/wgs84pos#long"

and are well-defined by an exterior ontology. Additionally, a client gets the signal that the specified Hypercat JSON-LD based catalogue can support the geographic bounding box search, only with the following information:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#GeoboundSearch"}

The following parameters are supported by the geographic search:

i. "*geobound-minlat*", which represents the lower limit of the latitude of the bounding box.
ii. "*geobound-maxlat*", which represents the higher limit of latitude of the bounding box.
iii. "*geobound-minlong*", which represents the lower limit of longitude of the bounding box.
iv. "*geobound-maxlong*", which represents the higher limit of longitude of the bounding box.

Geographic search can be executed by presenting the above-mentioned parameters.

Hypercat Lexicographic Range Search: The lexicographic search gives the permission and the opportunity to search for items which, (when organised lexicographically), fall among a minimum and maximum. Additionally, a client gets the signal that the specified Hypercat JSON-LD based catalogue can support the lexicographic range search, only with the following information:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#LexrangeSearch"}

Moreover, the property "*hypercat:lastUpdated*" is available for dates and time. The following parameters are supported by the lexicographic search:

i. "*lexrange-p*", which represents the JSON-LD's property to search on.
ii. "*lexrange-min*", which represents the lower limit of range to return (inclusive).
iii. "*lexrange-max*", which represents the higher limit of range to return (non-inclusive).

Lexicographic search can be executed by presenting the aforementioned parameters.

Hypercat Robots Exclusion Search: A Hypercat JSON-LD catalogue has also the ability to provide information to the client for a related "*robots.txt*" file, by applying the property "*hypercat:hasRobotstxt*". This can be used by websites to connect with web crawlers and additional web robots. Take notice that, if a catalogue has a robots exclusion document, then the document must be placed at the BASE URL with the name "*robots.txt*", and the value (JSON-LD's object) must be a URL with the following form "*\[BASE URL]/robots.txt*".

Hypercat Multi-Search: The Hypercat has the ability to support numerous search extension types. These extensions are mostly alternative forms of the simple search, and they can only permit for simple connections with the catalogue. In order to combine geographic search and lexicographic range search two independent queries can be submitted and their results can be merged manually by the client, resulting in an inefficient solution. The multi-search lets the client to put together simple or multi search mechanisms and capture the wanted items. Additionally, a client gets the signal that the specified Hypercat JSON-LD based catalogue can support the multi-search, only with the following information:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#MultiSearch"}

A multi-search query takes a single JSON object. The following parameters are supported by the multi-search:

i. "*query*", which represents a JSON object, which holds a URL query string as passed to underlying search mechanism,
ii. "*intersection*", which represents a JSON array of objects containing query, intersection or union.
iii. "*union*", which represents a JSON array of objects containing query, intersection or union.

All searches can be nested to allow a complex combining of intersection and union.

Hypercat Prefix Match Search: The prefix search permits the search for items where the search parameter is a prefix match of a catalogue item. Additionally, a client gets the signal that the specified Hypercat JSON-LD based catalogue can support the prefix match search, only with the following information:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#PrefixSearch"}

If there are several search parameters, the server must return the connection of items where search parameters match with a single item, mixing parameters with boolean AND. The following parameters are supported by the prefix match search:

i. "*prefix-s*", which represents a prefix of the JSON-LD's subject.
ii. "*prefix-p*", which represents a prefix of the JSON-LD's property.
iii. "*prefix-o*", which represents a prefix of JSON-LD's object.

Hypercat Linked Data rel: To define that the Hypercat JSON rel item is an instance of RDF class, the keyword *@type* must be added. Thus, no new property needs to be added.

Hypercat License rel: Hypercat catalogues or linked resources can be accessible under a precise license. Moreover, the property "*hypercat:hasLicense*" inside a catalogue lets the clients to specify the license under which the data is released. This property must show a machine or human understandable form of the license. When there are numerous "*hypercat:hasLicense*" declarations, the client should expect that there are numerous licences for the specified resource.

Chapter 6

Semantic Search

Semantic search lets the client use SPARQL-like queries to the semantically enriched Hypercat catalogues, delivering a search type that detects the underlying semantic hierarchy. As mentioned above, JSON-LD can support every RDF major concept that's why SPARQL can be used to query every JSON-LD based catalogue. Any valid JSON-LD catalogue is semantically enriched, thus supporting queries that return all catalogue items belonging to a specific class or its subclasses. When a query has the *rel* (or JSON-LD predicate) which is "*@type*" (rdfs:type) and *val* (or JSON-LD object) which is "*bt-hypercat:SensorFeed*", and with the BT catalogue's ontology mapping that the "bt-hypercat:SensorFeed" is a subclass of "*bt-hypercat:Feed*", semantic search will return all catalogue items that are instances of the two "*bt-hypercat:Feed*" and "*bt-hypercat:SensorFeed*". For example, it can return the "*bt-hypercat:belongsToSensorFeed*", "*bt-hypercat:feed_title*", "*bt-hypercat:feed_tag*" of each catalogue items and help the searcher find relatively items with the same "*bt-hypercat:SensorFeed*" of the given query. This would not be able to be done without the use of the Hypercat JSON-LD to code the subclass ontology and work with it. Additionally, a client gets the signal that the specified Hypercat JSON based catalogue can support the semantic search, only with the following *rel*, *val* information:

"rel": "urn:X-hypercat:rels:supportsSearch"
"val": "urn:X-hypercat:search:semantic"

whilst a JSON-LD based catalogue should contain the following:

"@id": "http://portal.bt-hypercat.com/cat-json-ld",
"http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch": {
"@id": "http://portal.bt-hypercat.com/ontologies/hypercat#SemanticSearch"}

If there are many search parameters, the server has to put these parameters into a single triple pattern and run a SPARQL-like query, with reasoning based on the both catalogue's ontology and catalogue itself. The following parameters are supported by the Semantic search for JSON-based and JSON-LD based catalogues:

i. "*sem-href*" ("*sem-s*" for JSON-LD), which represents the resource URI (the subject for JSON-LD).

ii. "*sem-rel*" ("*sem-p*" for JSON-LD), which represents the semantic metadata relation (the predicate for JSON-LD).

iii. "*sem-val*" ("*sem-o*" for JSON-LD), which represents the semantic metadata value (the object for JSON-LD) (Tachmazidis et all, 2016).

The Semantic search can be important and a useful addition to the Hypercat catalogue. This kind of search aims to accurate and make better the search experience of the user by understanding the user's intention on a given query (Lashkari et al, 2017). This search type can find relative items to the one the user search for and show them to the results. For Hypercat Semantic search can also find relative *sensors*, *datasreams* and *events* to the one the user search for, and then show a results list not only of the item the user search and its meta-data but also every single item and its meta-data which has a relative connection. For example, searching for any *datastream* in the Hypercat catalogue:

?sem-p=@type&sem-o=http://portal.bt-hypercat.com/ontologies/bt-hypercat#Datastream

the result will not only show items of type Datastream but also any subclass of the class Datastream, namely it will get items with "@type":

"@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"
"@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#EventStream"

and return any item matching the search criteria:

{"@id":"http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/0",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" : {
"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_max_value":"max_value=50",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_min_value":"min_value=-30",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "degrees Celsius",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol":"degrees Celsius",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type": "derivedSI",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "temperature",
"@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"},
{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/1",

```
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :  {
 "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2" },
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text":"16point compass",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type":  "contextDependentUnits",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "winddirection",
"@type" : "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"},
{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/2",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :  {
 "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "miles per hour",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol": "mph",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type": "derivedUnits",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag": "windspeed",
"@type": "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"},
{"@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2/datastreams/3",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#belongsToSensorFeed" :  {
 "@id": "http://api.bt-hypercat.com/sensors/feeds/fa899552-044a-48e2-bdbe-0076292fd4a2"},
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_text": "miles per hour",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_symbol": "mph",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_unit_type": "derivedUnits",
"http://portal.bt-hypercat.com/ontologies/bt-hypercat#datastream_tag" :  [
      "windspeed",
      "gust"],
"@type" :  "http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorStream"}
```


The implementation of the semantic search has not been tested yet in a real server because of the lack of time, but the following paradigm of the design, shows how a semantic search can work with a Hypercat JSON-LD based catalogue. The MongoDB is used to achieve SPARQL-like queries of JSON data and the reasoning of the ontology can be achieved applying the reasoners Pellet and HermiT. The Hypercat JSON-LD catalogue can be stored into a MongoDB following the instructions of (Kellogg, 2012) telling that MongoDB can work and query perfectly with JSON-LD because MongoDB document model is a normal fit for JSON-LD. The Figure 9 shows the Semantic search workflow from

the JSON to JSON-LD, then following the ontology to the reasoners additionally from the reasoners to Mongo-DB and at the end to the front-end webpage.

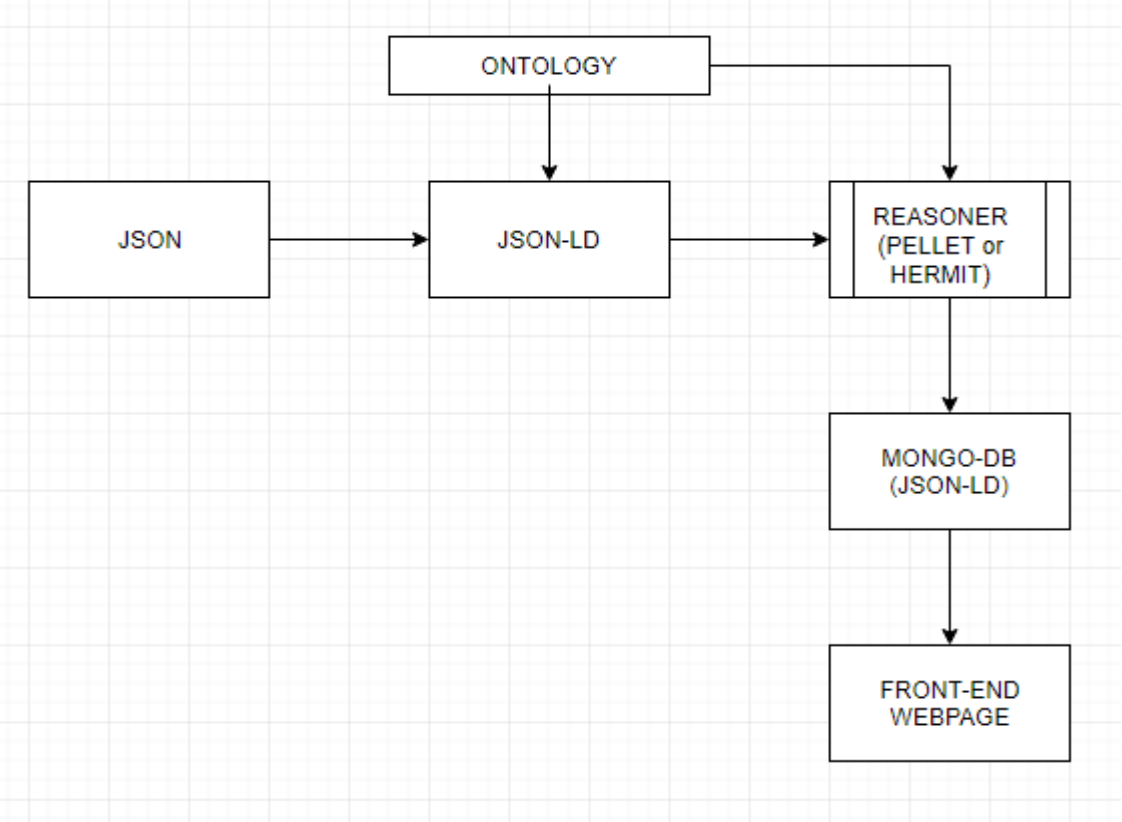

Figure 9: Semantic Search Implementation's Design

Another paradigm of the semantic search is once the JSON-LD and the Ontology pass from the reasoner the JSON-LD will be stored into the MongoDB and will stay there to be queried. There will be no more reasoning after the first time because then it will get all the details and information from the subclass hierarchy. In addition, when a user gives a query then the query will go only to the database and then return with the results in the front-end webpage. Thus, for a given query where the user searches for all of the Datastreams into the catalogue, the Semantic search through MongoDB it returns not only the directed Datastreams, but all of the derived Datastreams (through reasoning), such as SensorStream and EventStream.

Chapter 7

Conclusion

In this thesis, a semantic enrichment for the Hypercat specification has been presented, which allows the description of a JSON-LD based catalogue. Moreover, it was displayed how existing JSON-based catalogues can be translated into JSON-LD based catalogues in an automatic way. It is worth noting that, the Hypercat system its written in JSON language so it is easier for a JSON developer to assimilate and translate or even create the catalogue to JSON-LD and having all the semantic features, instead of translating it to any other language. The challenge was to create a Semantic Hypercat system which has the ability of high interoperability, without losing everything from the main Hypercat specification and JSON syntax. Additionally, to enchase the catalogues with semantic search and to complete the whole translation from a JSON-based catalogue to JSON-LD based catalogue with the minimum requirements, such as removing the ‘context’ from the JSON-LD and develop the catalogue in a more extended form. In future work, there is a plan to standardize the Hypercat JSON-LD specification by working closely with the Hypercat community and apply it in the world. There will be also an implementation of the Semantic search mechanism with real IoT data in a real Datahub, and an investigation of how Semantic Hypercat can be more impactful in any company or organization. In this way, rich Hypercat catalogues will deliver a high degree of interoperability.

References

Antoniou, G. & Harmelen, F. v. (2004). *Web Ontology Language: OWL*. 67-92.

Antoniou, G. & Harmelen, F. v. (2008). *A semantic web primer (2nd ed.)*. Cambridge, Mass: MIT Press.

Beart, P. (2016). *Hypercat 3.00 Specification.*

Bellini, P., Nesi, P., & Rauch, N. (2014). *Smart City data via LOD/LOG Service.* Retrieved from https://www.researchgate.net/publication/262487273_Smart_City_data_via_LODLOG_Service.

Bradley, A. (2014). *What is JSON-LD? A Talk with Gregg Kellogg*. Retrieved from http://www.seoskeptic.com/what-is-json-ld/.

d'Aquin, M., Adamou, A., Daga, E., Liu, S., Thomas, K., & Motta, E. (2014). *Dealing with Diversity in a Smart-City Datahub.* 68-82.

Davies, J. & Fisher, M. (2015). *Internet of things - Why now?*. 9. 35-42.

Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. (2013). *Internet of things (IoT): A vision, architectural elements, and future directions. Future Generation Comp. Syst.*, 29(7), 1645-1660.

Ismail, S., & Shaikh, T. (2016). *A Literature Review on Semantic Web–Understanding the Pioneers' perspective.* In The Sixth International Conference on Computer Science, Engineering and Applications (Vol. 6, No. 11, pp. 15-28).

JSON. (N/A). *Introducing JSON* . Retrieved from http://www.json.org/.

Kellogg, G. (2012). *JSON-LD and MongoDB*, NoSQL Now! 2012, San Jose, CA, USA. Retrieved from https://www.slideshare.net/gkellogg1/jsonld-and-mongodb.

Lanthaler, M. & Gült, C. (2012). *On Using JSON-LD to Create Evolvable RESTful Services.* In Proceedings of the Third International Workshop on RESTful Design, Lyon, France, Retrieved from https://json-ld.org/learn.html.

Lashkari, F., Ensan, F., Bagheri, E., & Ghorbani, A. A. (2017). Efficient indexing for semantic search. Expert Systems with Applications, 73, 92-114. doi:10.1016/j.eswa.2016.12.033

Lecue, F., Tucker, R., Bicer, V., Tommasi, P., Tallevi-Diotallevi, S., & Sbodio, M,L.(2014). *Predicting Severity of Road Traffic Congestion Using Semantic Web technologies.* 611-627.

Murelo, R., Almeida, A., Azkune, G., Mainetti, L., Mighali, V., Patrono, L. ... Sergi, I. (2017). *An AAL system based on IoT technologies and linked open data for elderly monitoring in smart cities.* In 2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech), Split, Retrieved from https://morelab.deusto.es/media/publications/2017/conferencepaper/an-aal-system-based-on-iot-technologies-and-linked-open-data-for-elderly-monitoring-in-smart-cities.pdf.

Pauwels, P., Zhang, S., & Lee, Y. (2017). *Semantic web technologies in AEC industry: A literature overview.* Automation in Construction, 73, 145-165. doi:10.1016/j.autcon.2016.10.003

Phuoc , D. & Hauswirth, M. (2009). Linked Open Data in Sensor Data Mashups. In *Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN09). 522.* , Retrieved from https://www.researchgate.net/publication/230745492_Linked_Open_Data_in_Sensor_Data_Mashups .

Sporny, M. (2014, January 21). JSON-LD and Why I Hate the Semantic Web [Web log post]. Retrieved from http://manu.sporny.org/2014/json-ld-origins-2/.

Stolz, A., Rodriguez-Castro, B., & Hepp, M. (2013). *RDF Translator: A RESTful Multi-Format Data Converter for the Semantic Web*. Retrieved from http://www.stalsoft.com/publications/rdf-translator-TR.pdf.

Su, X., Riekki, J., Nurminen, J. K., Nieminen, J., & Koskimies, M. (2015). *Adding semantics to internet of things*. Concurrency and Computation: Practice and Experience, 27(8), 1844-1860. doi:10.1002/cpe.3203

Tachmazidis, I., Davies, J., Batsakis, S., Antoniou, G., Duke, A., & Stincic Clarke, S. (2016). *Hypercat RDF: Semantic Enrichment for IoT.*

Tachmazidis, I., Davies, J., Batsakis, S., Antoniou, G., Duke, A., & Stincic Clarke, S. (2017). *A Hypercat-enabled Semantic Internet of Things Data Hub: Technical Report.*

The Linked Open Data Cloud. (2018). *About the diagram*. Retrieved from https://lod-cloud.net/.

W3C. (2012). *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Retrieved from https://www.w3.org/TR/owl2-overview/.

W3C. (2014). *RDF 1.1 Turtle Terse RDF Triple Language*. Retrieved from https://www.w3.org/TR/turtle/.

W3C. (2017). *JSON-LD Primer.* Retrieved from https://json-ld.org/primer/latest/.

W3C. (2018). *JSON-LD 1.1 A JSON-based Serialization for Linked Data.* Retrieved from https://w3c.github.io/json-ld-syntax/.