ETSI GS CIM 006 V1.2.1 (2023-06)

GROUP SPECIFICATION

Context Information Management (CIM); NGSI-LD Information Model

Disclaimer

The present document has been produced and approved by the cross-cutting Context Information Management (CIM) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGS/CIM-006v121

Keywords
information model, interoperability, smart city

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
https://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

If you find a security vulnerability in the present document, please report it through our Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) cross-cutting Context Information Management (CIM).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The purpose of the present document is to give property graphs a formal semantic grounding based on RDF/RDFS/OWL, with blank nodes reification, geared to JSON-LD serialization. On top of it, a set of core cross-domain ontology classes have been defined, based on this meta-model. This whole information model is meant to be used by many applications as a basis for data representations. It is compatible with the NGSI-LD API defined in ETSI GS CIM 009 [2].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] W3C® Recommendation 16 July 2020: "JSON-LD 1.1: A JSON-based Serialization for Linked Data".

[2] ETSI GS CIM 009 (V1.7.1): "Context Information Management (CIM); NGSI-LD API".

[3] W3C® Candidate Recommendation Draft 15 November 2022: "Time Ontology in OWL".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Guinard, D., & Trifa, V. (2016): "Building the web of things", Shelter Island: Manning.

[i.2] Tim Berners-Lee (2006-07-27): "Linked Data", Design Issues W3C.

[i.3] J. Frey, K. Müller, S. Hellmann, E. Rahm and M.-E. Vidal (2017): Semantic Web - Interoperability, Usability, Applicability an IOS Press Journal: "Evaluation of Metadata Representations in RDF stores".

[i.4] Cassandras, C. G., & Lafortune, S. (2009) Springer Science & Business Media: "Introduction to discrete event systems".

[i.5] W3C® Editor's Draft 11 August 2005: "Simple part-whole relations in OWL Ontologies".

[i.6] W3C® Working Group Note 9 March 2006: "A Semantic Web Primer for Object-Oriented Software Developers".

[i.7] W3C®: "HttpRange14Webography".

[i.8] W3C® Interest Group Note 3 December 2008: "Cool URIs for the Semantic Web", Leo Sauermann and Richard Cyganiak.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

cross-domain ontology: part of the information model that defines generic classes (formal concepts and constructs, with associated constraints) that serve as common denominators between domain specific models, addressing the temporal and structural description of physical systems

domain-specific ontologies: information models that define base classes and their constraints, within specific technical domains (e.g. buildings, transportation, agriculture) and define their structure and vocabulary

meta-model: part of the information model that formally defines the NGSI-LD foundational classes (Entities, Relationships, Properties and reification constructs) on the basis of RDF/RDFS/OWL

NGSI-LD entity: informational representative of something that is supposed to exist in the real world, physically or conceptually. Any instance of such an entity shall be uniquely identified by a URI, and characterized by reference to one or more NGSI-LD Entity Type(s)

NGSI-LD property: description instance which associates a main characteristic, which is an NGSI-LD Value, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property

NOTE: It includes the special "hasValue" property to define its target value.

NGSI-LD relationship: description of a directed link between a subject which is either an NGSI-LD Entity, an NGSI-LD Property, or another NGSI-LD Relationship on one hand, and an object, which is an NGSI-LD Entity, on the other hand

NOTE: It includes the special "hasObject" property to define its target object.

NGSI-LD value: JSON value (i.e. a string, a number, true or false, an object, an array), **or** a JSON-LD typed value (i.e. a string as the lexical form of the value together with a type, defined by an XSD base type or more generally an IRI), **or** a JSON-LD structured value (i.e. a set, a list, a language-tagged string)

EXAMPLE: An NGSI-LD Entity of type (Type Name) "Vehicle" (when parked) can be the subject of an NGSI-LD Relationship which has as its object a NGSI-LD Entity of type "Parking".

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
CIM	Context Information Management
CNIT	Consorzio Nazionale Interuniversitario per le Telecomunicazioni
DBMS	DataBase Management System
GSMA™	GSM Association
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IRI	Internationalized Resource Identifier
ISG	Industry Study Group
JSON	JavaScript Object Notation

JSON-LD	JavaScript Object Notation for Linked Data
LD	Linked Data
LOD	Linked Open Data
NGSI	Next Generation Service Interfaces
NIR	Non-Informational Resource
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
OWL-DL	Web Ontology Language - Description Logic
PG	Property Graph
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SAREF	Smart Applications REFerence ontology
SAS	Société par Actions Simplifiée
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
WoT	Web of Things
XML	eXtended Markup language
XSD	XML Schema Definition

4 Rationale for a multi-layered and multi-scale graph-based context information model

4.1 Why use a graph-based model?

Systems and environments about which context information is stored and managed encompass many physical and non-physical entities. Context comprises all characteristics of these entities, as well as their states and other dynamic properties, together with relationships that stand for real-world connections, i.e. physical, virtual, or even abstract connections between them. This context information may be consolidated on the basis of data obtained from many different primary sources and infrastructures. Typical examples of such systems would be smart homes, buildings, or cities. Such systems, due to the wide range of requirements and granularities, are complex from the semantic, structural and behavioural viewpoints.

The expressivity and versatility of graph-based models allows to bring the whole corpus of graph theory to bear and to capture key information about such complex environments, in a directly usable way, as the graph matches all kinds of real-world connections between different physical and non-physical entities.

Graph models bring a fresh view on the definition of context information. In the first wave of context-awareness research dating back to the early 2000s, context used to be mostly, and implicitly, user-centric, typically capturing e.g. the activity or location of mobile users to adapt services offered to them. A widely publicized definition of context, dating from this stage of research, was "any information that can be used to characterize the situation of an entity". In a broader view of context where the very notion of context is de-centred and relative, this definition may in fact remain valid if entities are represented as the nodes of a graph. Rather than through a vague notion of situation, context is defined in the present document as the set of properties characterizing these nodes, together with the set of relationships that enmesh them together, and the properties of these relationships. In this perspective, the primary data of one application may be the context of another, and vice versa. Context is, though decentered and broadly defined, the graph itself. NGSI-LD thus maintains and exposes context information as a graph of matching links between the informational units corresponding to real-world entities of these environments.

Traditional (mono-centred) context fitted rather well a classical object-oriented or key-value description, with a set of more or less detailed context features attached to a single entity. The multi-centered notion of context address in the present document requires breaking this rigid hierarchical model by using a more expressive, flexible and adaptable information model. Graphs are the only model adapted to capture the complex structure of inter-entity relationships that make up context information in the sense in which it is defined it here. This information need not be semantically defined from the outset: it may be natively structural information, capturing e.g. containment or adjacency relationships. The semantics of this context may be added in a later stage of graph enrichment. This model fits the natively distributed nature of context data sources.

The Web of Things (WoT) [i.1] does also involve a graph of sorts, but it dispenses with maintaining it explicitly inside a database. The WoT graph is, like the graph of the original web, an implicit 100 % distributed graph of hyperlinks, not between web pages but between resources corresponding to connected devices that expose an interface on the web (e.g. using HTTP, WebSockets, etc.).

This view also aligns well with the grand evolution of the Web towards Linked Data, an evolution proposed by W3C from the Semantic Web project [i.2] that is currently supported by RDF-derived graph models. Linked Data provides a method of publishing structured data so that it can be interlinked and support semantic queries.

4.2 Separating semantic referencing from structural descriptions

The NGSI-LD information model separates semantic referencing, used in the classical sense of the Semantic Web, from the structural description proper. The structural description may itself be decomposed into a basis structural graph whose nodes are physically-matched entities, and an overlay layer used to capture the way in which these entities are clustered into subgraphs.

Semantic referencing used by NGSI-LD is based on standard RDF/RDS/OWL typing and public ontologies, as shared by all other semantic information models. All nodes and edges of the structural graph are thus matched to several relevant classes/categories of these ontologies that jointly characterize the features shared by all instances of these classes.

A structural graph is a model of the structural description of an environment, capturing the relationships between the different subsystems that make up this environment. This description is, to some extent, independent of the overlaying semantic referencing, and it could be considered to "stand on its own", even without this referencing. A structural graph does in fact have a different kind of semantics of its own, such as e.g. when a graph captures and matches the structure of a physical network like a power grid or a water distribution network. These semantics apply to the graphs as a whole and are not reducible to the kind of "per-resource" semantics, which RDF is meant to describe.

4.3 Graph Examples used in the present document

Two examples of structural representations of city environments will be used as lead examples throughout the present document and are presented in Figure 1 Property graph example (1) and Figure 2 Property graph example (2).

The following graphical conventions are used throughout the present document:

- Regular Entity Regular (physically-matched) entities are represented as black rectangles.
- Relationship Relationships between these entities are represented as diamonds (rhombuses) overlaid on the corresponding arc of the graph, a convention borrowed from "entity-relationship" diagrams.
- Property Properties are represented by ovals that are on an arc between their entity or relationship and the property value, but often the arc is shortened to zero length for compactness.
- Value Values are represented as hexagons that may about the oval of the property of which they are the target, omitting an arc between the two.

Figure 1 describes a parking scenario, adjacent to two different streets. Information about the streets, parking places, and the sensors that monitor are attached to entities as shown in the figure. This example is intended to illustrate the full expressivity of a property graph as used to capture not only pure semantics, as an RDF graph would, but also structural and behavioural (in this case, the real-time state) information.

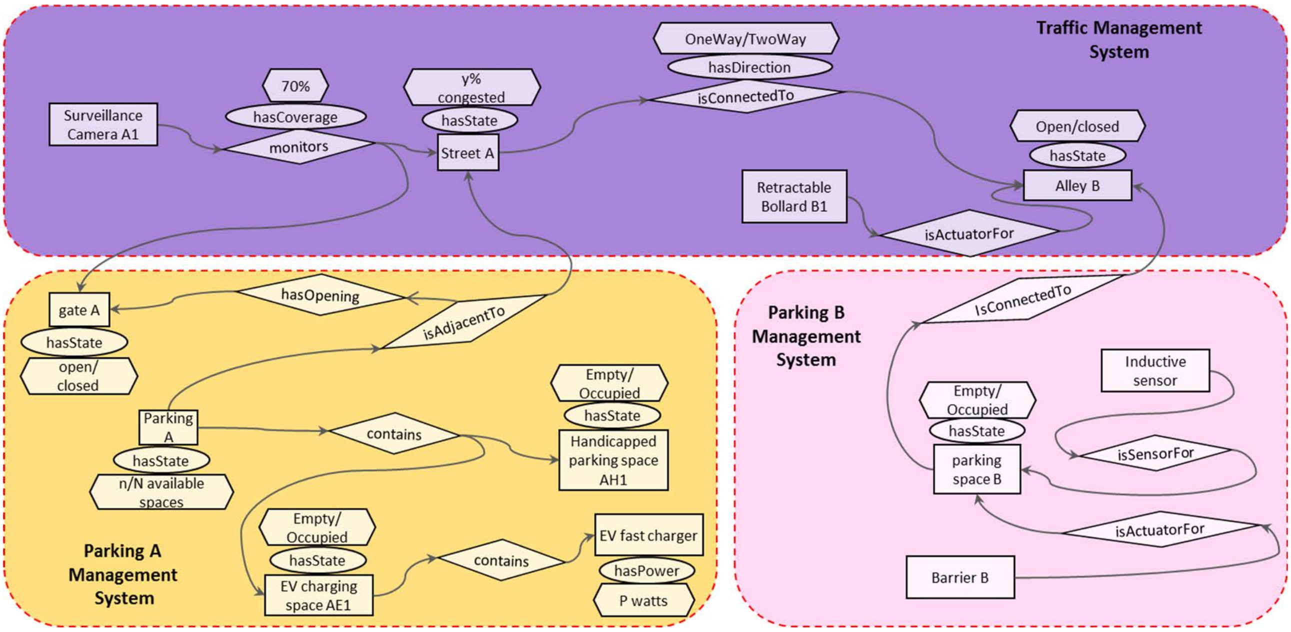

Figure 1: Property graph example (1)

Figure 2 example (2) is a more complex example used to illustrate intersecting domains and intertwined technical systems. The example consists of a building and its parts (using "hasPart" relationships) forming the structure of the building, in addition to other technical systems that are included in the building. The building is comprised of a garage and apartments (only one instance is represented below). A parking place within the garage belongs to the apartment, thus forming one system together. The building is equipped with a security system containing security devices. Additionally, there is a separate public parking that also appears in the example.

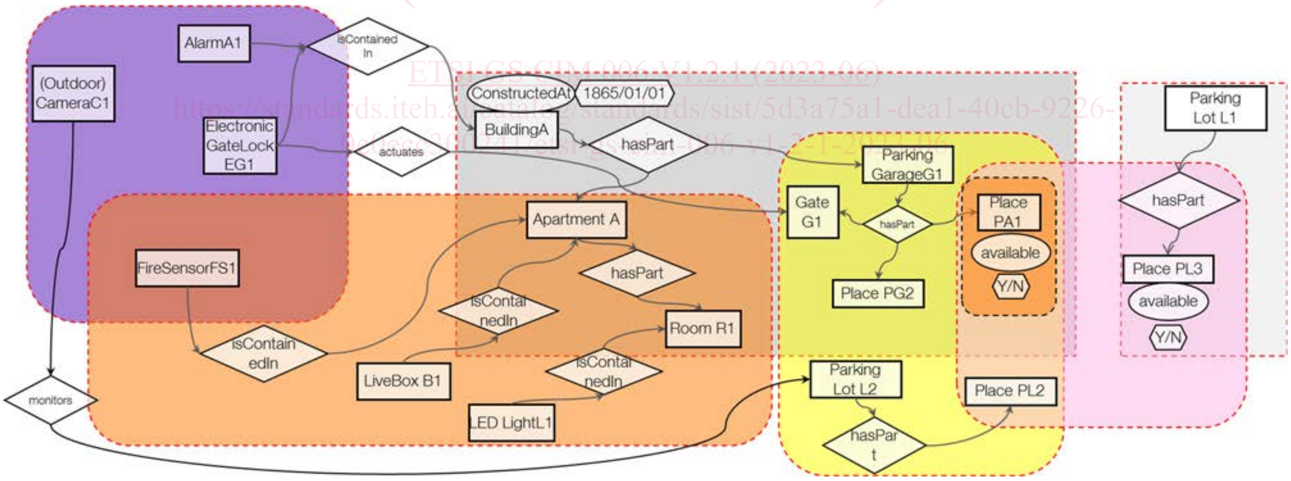

Figure 2: Property graph example (2)

5 NGSI-LD meta-model

5.0 Introduction

The NGSI-LD meta-model provides a formal basis for representing "property graphs" using RDF/RDFS/OWL. It makes it possible to perform back and forth conversion between datasets based on the property graph model on the one hand and linked data datasets which rely on the RDF framework, on the other hand. This may be seen as raising the semantic expressivity of RDF triples to the level of property graphs. Property graphs may, contrary to RDF, use predicates as subjects of other predicates (properties of properties and properties of relationships).

5.1 Fundamentals of property graphs and graph databases

Property graphs are the implicit semi-formal data models underlying most present-day graph databases. They have gained widespread following, more in industry than in academia. They make it possible to attach properties (defined as key-value pairs) to relationships, a feature which RDF does not directly support, but they lack the standardization and formal underpinnings of RDF and do not interoperate directly with linked data and other RDF datasets. Also they do not lend themselves to reasoning with RDF-based reasoning tools or querying with standard query languages such as SPARQL.

Property graphs are usually defined (informally) as follows:

- A property graph is made up of nodes (vertices), relationships, and properties.
- Nodes may have properties in the form of arbitrary key-value pairs. Keys are strings and values are arbitrary data types.
- A relationship is an arc (uni-directional, i.e. directed edge) of the graph proper, which always has an identifier, a start node and an end node. Like nodes, relationships can have properties attached to them.

There are several key differences between property graphs (PG in the following) and RDF graphs:

- RDF properties are expressed as regular triples, i.e. arcs of the graph with start node and end node, and their target can be either a literal, an IRI or a blank node [1], whereas the target of a PG property always corresponds to an RDF literal.
- PG relationships (i.e. primary graph links between PG vertices) are first-class citizens of the PG model and have an internal structure similar to that of a vertex, inherited from object-oriented modelling object, with an optional set of properties defined by key-value pairs.
- The distinction between relationships and properties in the PG model is similar to the distinction between object properties and datatype properties in OWL, but stronger.
- PG properties are, for simplicity and avoiding clutter in diagrams, usually not represented as additional arcs of an underlying graph, but are represented as attached to vertices or relationships.
- Identifiers in a Property Graph need only be unique within the scope of a given graph (typically as internal identifiers assigned by the Graph DBMS), and need not be universally unique like URIs/IRIs.
- Property graphs can be queried with graph-specific query languages that may use graph patterns (complete subgraphs) as query terms, i.e. are not limited to only nodes identified with specific key/values.
- PG properties and relationships are individually identified when instantiated, whereas RDF properties are not instantiated nor identified as individual resources, being only defined by their property type.
- Properties cannot be directly attached to the arc (predicate) of an RDF triple, but RDF reification makes it possible, in several different ways, to circumvent this limitation by turning a triple into a resource.

5.2 Reification with blank nodes

In the RDF formalism, the *reification* of a statement turns it into a resource, so that it can be the subject of another statement. Making statements about statements is useful e.g. for providing information about the provenance (lineage) of data. It is indispensable for transforming a property graph into an RDF dataset. Many different reification solutions have been proposed. Reification by way of blank nodes is the simplest for the current purposes and is the solution chosen by ETSI ISG CIM. Consider the following simple example.

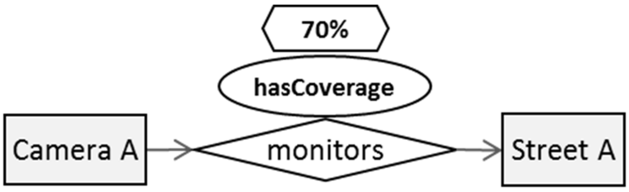

Figure 3: Property graph example to be represented in RDF using reification

To express that the camera monitors only 70 % of the street area, which obviously is not a property of the street, nor of the camera, but of their relationship, it is needed to reify this statement about the relationship:

[CameraA → monitors → StreetA]

in order to make it the subject of another statement:

[[statement_1] → hasCoverage → 70 %],

This can be done by adding a blank node to obtain an RDF-reified equivalent of the example property graph with three triples as follows and as visualized in Figure 4:

*[CameraA → monitors → **_blankNode_n**]*

*[**_blankNode_n** → hasObject → StreetA]*

*[**_blank_node_n** → hasCoverage → "70 %"]*

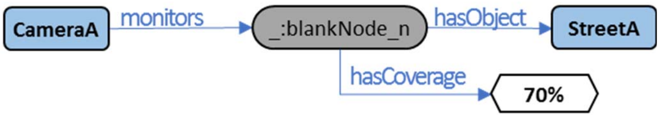

Figure 4: RDF reified example

This solution is especially convenient when the graph is serialized using JSON-LD ([1] see following clause) because blank nodes do not explicitly appear in the textual serialized description, and actually show up only when it is represented as a RDF graph. It is thus possible for a developer to generate the JSON-LD payload required by the NGSI-LD API in a form that is very similar to what he would have generated in plain JSON. The simplicity of JSON-LD representation of property graphs reified with blank nodes is a key argument behind the choice of this solution.

With alternative reification methods, users and developers shall include supplementary terms and shall deal with complex redundant terms that may distract and confuse them. Several such reification methods have been proposed in the literature (see e.g. [i.3]). For comparison, here is a brief description of three of the more widely used reification methods:

- *Classical RDF reification* defines a new RDF resource that is linked back to the original statement. This uses RDF built-in reification capabilities, as RDF natively provides a vocabulary intended for describing RDF statements, namely the type `rdf:Statement`, and the properties `rdf:subject, rdf:predicate, and rdf:object`. A total of 4 additional statements (corresponding to the so-called "reification quad") are required to fully define a statement as a resource, and this is just in order to be able to make this resource the subject or object of other statements.

- *Singleton properties:* this other simple solution to reification amounts to identifying each predicate instance individually as a resource with its own per instance IRI, and using this new resource as the subject of another statement. This actually changes the nature of the original RDF graph because what was originally an arc of the graph becomes a vertex of the transformed graph.

- *Named graphs/quads:* all RDF triples are redefined as "quadruples" ("quads" for short, unrelated to the "reification quads" mentioned above). These quads are generalizations of triples, with 4-fold *arity* (whereas the above "reification quads" are sets of 4 triples!). These quads have as the extra element an associated IRI that identifies the statement as an instance of the corresponding predicate (which is itself, as per the RDF model, defined by a generic IRI). This fourth element, the IRI, makes it possible for the statement to be the subject (or object) of another RDF triple. Named graphs are much more powerful than this basic quad mechanism, in that they allow any subgraph (set of interconnected triples) to be jointly identified. JSON-LD [1] supports these general named graphs but it is a very cumbersome and heavyweight means to reify simple triples. The reification method used in the present document is much more lightweight than the "*Named graphs*" approach.

These reification methods are compared in Figure 5 (Standard RDF reification - with quads) and in Figure 6 (Singleton property reification), with the proposed blank-node-based reification method, from two points of views: (a) JSON-LD corresponding representation and (b) SPARQL query complexity for extracting data.

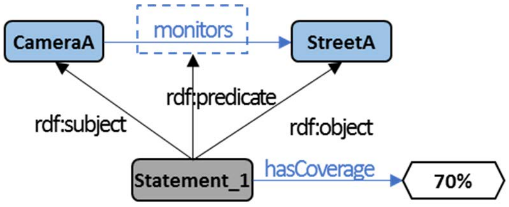

Figure 5: Standard RDF reification

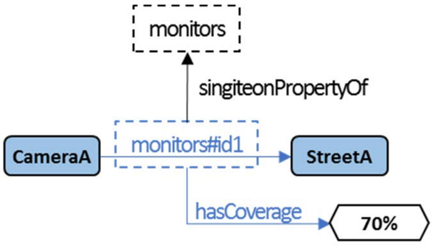

Figure 6: Singleton property reification

JSON-LD Format:

```
[
{"@id": "CameraA",
"monitors": {"@id": "StreetA"}}

{"@id": "Statement_1",
"subject":
 {"@id": "CameraA"},
"predicate":
 {"@id": "monitors"},
"object":
 {"@id": "StreetA"},
"hasCoverage": "70%" }
]
```

SPARQL Query:

```
SELECT ?R WHERE {
?st rdf:subject  :CameraA.
?st rdf:predicate
                :monitors.
?st rdf:object StreetA.
?st :hasCoverage ?R
}
```

JSON-LD Format:

```
[
{"@id": "CameraA",
"monitors#id1":
 {"@id": "StreetA}}

{"@id": "monitors#id1",
"singletonPropertyOf":
 {"@id": "monitors"},
"hasCoverage":"90%" }
]
```

SPARQL Query:

```
SELECT ?R WHERE {
:CameraA ?p :StreetA.
?p :singletonPropertyOf
               :monitors.
?p :hasCoverage ?R
}
```

Using reification with blank nodes, the SPARQL query is as follows:

```
SELECT ?R WHERE {
:CameraA :monitors ?bn.
?bn :hasObject :StreetA.
?bn :hasCoverage ?R.
}
```

For targeting directly the query to the object of "monitors" instead of the value of the coverage of the monitoring, the `owl:propertyChainAxiom` is used as follows:

```
: monitors owl:propertyChainAxiom (:monitors:hasObject).
```

This can be defined for all reifiable properties similar to "monitors" in the preceding statement. The SPARQL query for the object of the property becomes simple and equivalent to queries without reification.

```
SELECT ?S where {
:CameraA : monitors ?S.}
```