Things Data Interoperability Through Annotating oneM2M resources for NGSI-LD Entities

Sunil Kumar[1], SeungMyeong Jeong[1], Il Yeup Ahn[1], and Muhammad Aslam Jarwar[2]

[1]*Autonomous IoT Research Center, Korea Electronics Technology Institute (KETI), #25, Saenari-ro, Bundang-gu, Seongnam-si, Gyeonggi-do 463-816, Korea*
[2]*Department of Computing, Sheffield Hallam University, Sheffield, UK*

sunil75umar@keti.re.kr, sm.jeong@keti.re.kr, iyahn@keti.re.kr, a.jarwar@shu.ac.uk

***Abstract*—In this era of Information and Communication Technology (ICT) and the Industrial Internet of Things (IIoT), semantic interoperability plays an important role in interworking among different standards. One such standard is oneM2M, which supports semantic interoperability between non-semantic oneM2M resource model and semantic data, but it is only limited to Resource Description Framework (RDF) triple data. Where Next Generation Service Interfaces – Linked Data (NGSI-LD) – provides information model and protocol for enhancing the capabilities to represent more complex structures of Linked Data, limited research has been conducted regarding such framework or protocol to support the interpretation and translation among these two different standards. This paper proposes a mapping protocol for interpreting and translating non-semantic oneM2M resource data to NGSI-LD interfaces.**

Keywords— mapping ontology, RDF based mapping, IoT data interoperability

I. INTRODUCTION

In the fourth industrial revolution, more challenges related to the Internet of Things (IoT) and distributed environment are being resolved using No Sequel databases, Graph models and Semantic Web technologies, etc. Graph databases have been considered viable in different research as well as enterprise solutions, yet there exist some key differences among various formats in which Resource Description Format (RDF), Web Ontology Language (OWL) and Labelled Property Graph (LPG) are widely used [1], [2]. The essential difference between RDF triple and LPG is their structural organization. The LPG has a complex structure where the relation or an edge connecting the two nodes can also have its properties (attributes) as well as can have another connection point through a relation, to be connected with a third node or a relation. Whereas in RDF triples (formed by the components - subject, predicated, and object), the relation (predicate) can only have a subject and an object as two connection points. Besides the structural differences, their applications also vary. RDF and OWL formalizations are leveraged, where semantic annotation, description logics, and reasoning are required, such as interworking services, taxonomy-based research, etc. Whereas LPG is preferred for complex structural representation and faster as well as complex graph analytic functions, as they provide compact storage space and serialization with higher graph analytics features.

The oneM2M standard, which is a global initiative for establishing M2M communication through a middleware architecture solution and protocols, has also extended its support for semantics through base ontology as well as integration schemes to map and add the RDF triples, either by referencing or by adding as a semantic descriptor resource [3]. It also supports advanced features such as reasoning, semantic mash-up as well as ontology mapping for interworking with different domain ontologies. However, semantic support in oneM2M is limited to RDF triples, and mechanisms to leverage the LPG knowledge graph are not sufficient.

NGSI-LD is a set of interfaces, developed by the Context Information Management Industry Specification Group of the European Telecommunication Standards Institute (ETSI ISG CIM), which established a hybrid approach, defining RDF-based vocabulary (NGSI-LD meta-model) [4] as the groundings for graph structures that can be mapped directly to LPG, allowing enhanced knowledge expressivity with efficient graph analytics. The meta-model is comprised of Entity, which represents a concept in the data; Properties, describing the characteristics of an Entity; Value, quantifying or qualifying the Properties; Relationship, connecting two Entities or even another Relationship. Using the meta-model, different domain models can also be extended, similar to domain ontologies in RDF and OWL.

Conversion of RDF to LGP graphs has been researched through different methods such as RDF statement or blank-node based NGSI-LD reification [4]. In that case, the translation of the oneM2M non-semantic resources to NGSI-LD-based LPG becomes a two-step process: first, annotating the resources into RDF triples; and, then, converting the triples into NGSI-LD Entities. This process is presumably tedious, complex, and still cannot guarantee a concise representation of annotated LPG data since it has been translated from triple data (edges with only two connection points).

Another possibility of translating oneM2M resources is through new protocol and resource specifications, which also must be aligned with the existing oneM2M specifications, similar to the specifications for RDF-based semantic support. However, the existing semantic support in oneM2M does not specify the exact annotation and translation techniques, as the standard is open to different oneM2M compliant RDF-based implementations, and does not declare a generalized annotation mechanism.

In this paper, we propose a mapping framework to bridge the gap between linked data support of oneM2M and NGSI-LD, by mapping the oneM2M resources to NGSI-LD concepts using RDF triples. Using the RDF mapping, it can be annotated into the NGSI-LD-based LPG. The resource management is handled at the data layer. These non-semantic resources can be translated to knowledge graphs at the upper layers where knowledge creation and extraction are performed

using NGSI-LD. This framework can be implemented in the existing oneM2M systems and does not require modification or addition of its specifications.

II. Related Work

There have been different efforts for mapping the IoT data, such as Amazon Web Service (AWS) IoT Things Graph [5] and Smart Geo Layers (SGeoL) [6], which have realized the need for interoperability among devices representing similar data, but in different formats. Both the AWS IoT Things Graph and SGeoL emphasize a uniform data model for achieving interoperability. SGeoL has been adapted by LGeoSIM [7], which is an NGSI-LD-based RDF Ontology, supporting further abstractions for different urban domains. Unlike SGeoL and AWS IoT Things Graphs, in LGeoSIM the applications extend the ontology model for their domain-specific requirements. Nonetheless, LGeoSIM requires data validation with its core ontology model.

Relational Database (RDB) to RDF Mapping Language (R2RML) [8], standardized by W3C, is a significant effort to map records in relational database to the instances in the RDF graph. This work is then extended by RML [9], supporting other data formats such as XML and JSON. Its potential has been realized by different researches, focusing on different domains including healthcare [10].

The NGSI-LD has been adopted by different frameworks and solutions. For the security domain, Gonzalez-Gil et al. [11] proposed DS4IoT ontology. For the proof of concept, they used search engine called IoTCrawler [12] and mapped DS4IoT ontology to NGSI-LD meta-model, as IoTCrawler utilizes it as a data exchange format. This straightforward mapping scenario is adequate if the graphs do not require LPG characteristics. However, it does not optimize the LPG characteristics to achieve complex and, very often, concise knowledge representation than RDF graphs.

Bauer et al. proposed a Morphing Mediation Gateway (MMG) [13] for interworking between different IoT platforms including FIWARE, oneM2M Z-Wave, and GS1. The data translation is a two-step process: source data is translated to internal data format (which is the NGSI context information model), and then to target data format. This modular approach allows each source data to have a dedicated translation module. In that period, NGSI specifications were not based on linked data. For translating oneM2M resources to NGSI format, pre-defined semantic annotations (as semantic descriptors in oneM2M) were required. In this case, mapping examples for different use-cases have been defined [14]. However, the exact algorithm for generalized mapping is not provided, as solution architecture is dependent on the individual one-to-one mappings. This work has been recently supported with Machine Learning based semi-automated solution to aid the human expert in simplifying the translation procedure [15]. However, the mappings are based on the similarities between source and target data, and the NGSI-LD is considered as the neutral format.

Most relevant work regarding annotating the IoT data to NGSI-LD has been proposed in Aquedücte [16], and FED4IoT [17]. Aquedücte aims at integrating heterogeneous IoT data to NGSI-LD, through REST API as well as through file upload. For the annotation, the supported mapping is required to be embedded either within the data or in the NGSI-LD @context. For mapping representation, different mapping attributes (as key-value pairs) have been defined, along with mapping for the GeoProperty type. However, it limits the mapping to a simple form of one-to-one mapping, and does not consider complex mapping possibilities which may require specific keys/sub-keys to be selected for their respective values/sub-values.

Fed4IoT [17] is an EU project aimed at IoT virtualization supporting data interoperability. Particularly they have proposed translation schemes between oneM2M and NGSI-LD with some consideration and specifications. Unlike MMG, they do not entirely rely on semantic descriptors to perform the mapping from RDF data to NGSI-LD. In addition, translation does not require additional intermediary internal data. Their core strategy is to traverse and interpret the labels in the data. In addition, they also specified mapping based on the oneM2M resource hierarchy involving Application Entity, Containers, and Content Instances. The mapping specifications are adequate in terms of translating the required information. However, they put tighter constraints on oneM2M resource definition considerations, whereas oneM2M specifications are very flexible in terms of resource usage. In addition, they require each NGSI-LD property and a relationship to be mapped as a separate container resource in oneM2M. In this case, the data size of oneM2M resource (representing a single NGSI-LD Entity) can immensely increase as each resource also includes its own meta information, specific to oneM2M protocol [18].

This work has proposed a mapping model and a protocol, to map a non-semantic oneM2M resources to NGSI-LD-based LPG. The proposed mapping considers to some extent, the complex structural representations in JSON values, based on the location of the relevant attribute. In addition, this protocol can be implemented in oneM2M system without modifying or restricting its existing specifications, using the existing oneM2M semantic support.

III. Mapping Considerations

Different oneM2M resource (hereby referred to as *"resource" or "resources"*) formats are allowed in oneM2M, including Extensible Markup Language (XML) and JavaScript Object Notation (JSON). For the scope of this work, JSON is considered the *resource* format, since it maintains a high level of commonality with JSON-LD, the serialization format for NGSI-LD. Nonetheless, different factors have to be considered between *resource* and NGSI-LD representation structure. First, the *resource* attributes and their format can vary based on the application domain and the schema it utilizes. For example, one domain may define local coordinates as key-value pairs, whereas the other may use an array. Second, the *resource* data may not include concepts that can be mapped directly to LPG. In this case, nodes and edges need to be identified and annotated. Finally, some level of semantics is required for identifying the usage scope of NGSI-LD concepts defined in the meta-model and contexts, which may not be available in the *resources*.

A. Mapping Considerations based on Semantic Descriptor Resource

In this approach, we leverage Semantic Descriptor Resource (SMD) to map its respective *resources*. The mappings in an SMD involve the RDF triples, specifying the *source elements* mapped to their respective *target elements*. The *source elements* represent the data defined as the JSON key-value pairs in oneM2M Container (CNT) and the oneM2M Content Instance resources (CIN). The *target*

elements represent the data in JSON-LD. Through the SMD, we utilize the underlying oneM2M semantic infrastructure to support interoperability with NGSI-LD.

To achieve simplicity in aligning the *resource* hierarchy with NGSI-LD concepts, a structure has been proposed: the top-level resource (CSE, AE, or CNT) [18] and their child resources should potentially represent a single NGSI-LD Entity. Following that resource hierarchy, the underlying data will potentially map to the NGSI-LD Properties and Values, belonging to that Entity, as well as the Relationships linked to other NGSI-LD Entities (the NGSI-LD concepts: Entity, Property, Relationship, and Value are hereby referred to as *Entity*, *Property*, *Relationship,* and *Value* respectively). In the case of CNTs, multiple *Entities* can be represented by the top-level CNTs, under a single AE or a CSE. These structural considerations are adapted from the work by Detti et al. [17].

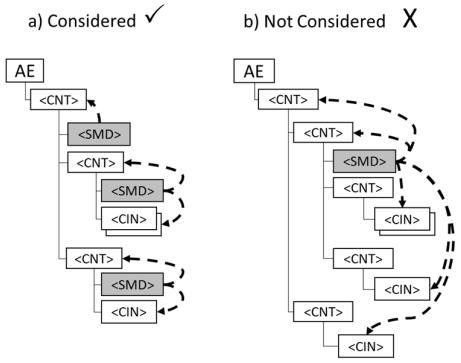

Fig 1. The placement and the resources mapped (indicated by red dashed arrows) by the respective SMD resource. a) Considered approach. b) The approach to be avoided.

In oneM2M, the SMD is used to annotate the parent *resource*, except when the parent is a CNT. In this work, SMD is assumed to provide the mapping for the direct sibling CIN, and the direct parent CNT in the *resource* hierarchy. In the case of multiple sibling CINs, a single SMD will be used. The reason for this restriction is to simplify the location and discovery of the appropriate SMD in the hierarchy. These considerations can be visualized in Fig. 1.

B. Mapping Ontology

To map the JSON-based key-value pairs to the concepts in NGSI-LD, a simple ontology has been defined, which can be used for the RDF mappings. Fig. 2 shows the ontology model.

This ontology is an extension of the NGSI-LD meta-model. The concept with prefix definition "ngsi-ld:" are the core concepts of NGSI-LD, whereas the ones with prefix definition *"annotation:"* represent the proposed mapping concepts. The properties *"sourceKey"*, *"targetKey"*, *"sourceValue"* and *"targetValue"* have key involvement in the mapping. Through these properties, the addresses or the values of *source elements* (*resource*s) can be mapped to their respective *target elements* (NGSI-LD data). In addition, an XML Schema Definition (XSD) datatype has been defined with URI *"annotation:reference"*. This is used to indicate that the *rdf:range* in the above-specified properties is an address to the key or value in the *source* or *target elements*. The property *"hasDomainType"* can be used to label an *Entity* type in a user-defined @context. The property *"targetValueType"* is used to specify a special type defined in NGSI-LD. Finally, the property *"hasEntityId"* is used to map the *Entity* identifier.

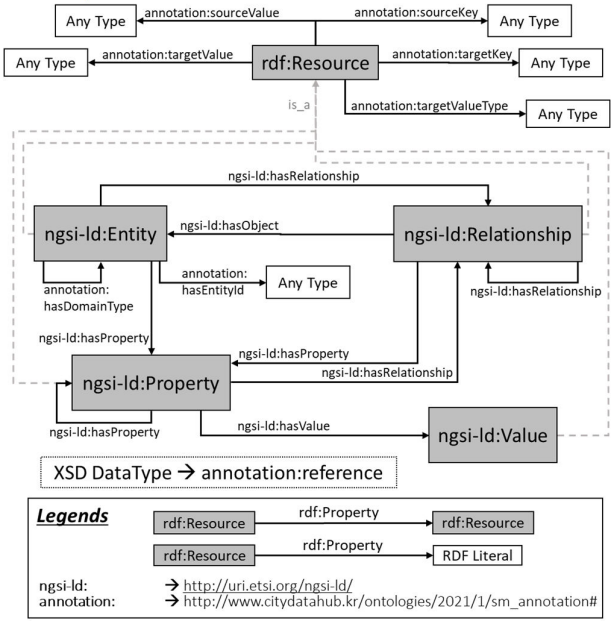

Fig. 2. RDF Mapping Ontology

IV. Mapping Generation and Interpretation

The mapping generation process involves a user who has the knowledge of both *resource* data and structure as well as the NGSI-LD. In a mapping application, the user manually selects the keys and values to be mapped to a single target *Entity* and its respective *Properties* and *Relationships*.

A. Mapping the NGSI-LD Entities and its Properties

Initially, the instance definitions of NGSI-LD concepts are defined by the user based on the given context. At the backend, the application will generate their respective instances of the *rdf:Resource* type. Then, for each of those instances, key and value mappings are defined using the RDF properties from mapping ontology. For the rdf:range of these properties, if the selected type is *"annotation:reference"*, then during the annotation, the application has to dereference the address, to locate the actual value in the data for the translation. Here two different addressing schemes have been adopted: one for *resources* and one for NGSI-LD. In oneM2M, "." (dot) is used for accessing the *resources* in the hierarchy. Similarly, while using the *"annotation:reference",* the "." followed by a name specifies the *sub-resource*. While accessing the content of the CIN, "." specifies the JSON keys and their respective values as objects to be accessed further. The starting point of the address is always the SMD itself in the hierarchy. To access the parent *resource* ".." (double dots) can be used. For addressing the NGSI-LD values, "/" (forward slash) has been used instead of ".". In case of NGSI-LD, the values will only correspond to the values of *Properties* or *Relationships* for a specific *Entity*. In both addressing schemes, there will be some cases where some part of the value is static, whereas another part has to be dereferenced. To distinguish between the two, the reference part has to be enclosed with "{}" (curly braces) inside which the same addressing scheme will be followed as defined above. In this case, the system treats the value as an

XSD String until it locates the address enclosed in "{}". Therefore "{}" are considered as reserved for this purpose.

Commonly, the *"sourceValue"* is not required, as the *"sourceKey"* can be used to access the value of that Key. However, when the source value needs to be translated as some complex JSON structure, then *"sourceValue"* or *"targetValue"* can be useful. The specification of properties defined in mapping ontology has been formalized in table 1.

TABLE I. MAPPING ONTOLOGY USAGE

rdf:Property	Usage
hasDomainType	Specify the type of the *Entity* instance as a direct/indirect subclass of ngsi-ld:Entity
hasEntityId	Specify the @type of *Entity* instance if its value is not in accordance to the required URI format
hasSourceKey	Specify an address or a value of a key, defined in the resource data for mapping the respective attributes of the *Entity*, *Relationship*, *Property* or a *Value*.
hasTargetKey	Specify an address or a value of a key, to be defined as a mapped attribute of the *Entity*, *Relationship*, *Property* or a *Value*.
hasSourceValue	Specify an address or a value, representing the value defined in the resource data for mapping the respective attribute value of the *Entity*, *Relationship*, *Property* or *Value*.
hasTargetValue	Specify an address or a value to be defined as a mapped attribute of the *Entity*, *Relationship*, *Property* or a *Value*.
targetValueType	Specify the explicit type (which cannot be determined from the *resource*) of the value to be defined as a mapped attribute of the *Entity*, *Relationship*, *Property* or a *Value*.

The complex relations in the NGSI-LD, such as *Property* of a *Property*, can be mapped using the properties defined in the NGSI-LD meta-model, such as ngsi-ld:hasProperty.

The instance representation of NGSI-LD concepts can be uniquely identified from their respective RDF resource URI in the mapping. However, mapping an instance of ngsi-ld:Entity is often not straightforward in a user-defined @context. For example, "ParkingLot" is defined as "Zone" and "Zone" is defined as ngsi-ld:Entity. In such case, if the instance in the mapping is specified as of type "ParkingLot", then the translation system will be unable to identify it as a ngsi-ld:Entity, and then the user-defined @context has to be accessed and interpreted for each translation. Therefore, we have defined such types using *"annotation:hasDomainType"* to retain the information for both ngsi-ld:Entity and a user-defined @context. Some information representing an *Entity* may not be available in the CIN, such as @Id, @type, etc. This information then must be available at the CSE, AE, or the top-level CNT, to be retrieved and mapped. This supports our assumption of SMD, mapping its direct parent *resource*. This mapping process is formalized using the following steps:

1) Locate the top resource representing NGSI-LD Entity.

2) Define the rdf:Resource of type ngsi-ld:Entity.

a) In case the type is specified in user-defined @context, define the type using rdf:Property "annotation:hasDomainType".

3) Locate the resource attribute and define the rdf:Property "annotation:hasEntityId".

4) Locate the data representing ngsi-ld:Property in the child CNT or CIN resources.

5) Define the rdf:Resource of the type as either ngsi-ld:Property or the one defined in the @context to represent the located attribute.

6) Link the previously defined rdf:Resource of type ngsi-ld:Entity with the above-defined rdf:Resource of type ngsi-ld:Property using ngsi-ld:hasProperty.

7) Locate resource attributes and define the mappings using annotation properties from Mapping Ontology.

8) If the ngsi-ld:Property is linked with another ngsi-ld:Property, then perform step 4 to step 7 with a small variation in step 6: instead of ngsi-ld:Entity, the two rdf:Resource of type ngsi-ld:Property will be linked.

9) Repeat step 5 to step 9 for the rest of the properties.

B. Mapping the Relationships among NGSI-LD Entities

Specifying the *Relationships* is complex due to the following reasons: the system may not be able to identify the relations in the *resources* as most often they are not structured like a graph; *Entities* have *Relationships* with other *Entities*, represented by *resources* located far away in the system. Such relations will require an additional discovery process in the oneM2M system; the *resources* may not exist all at once; defining such *Relationships* requires updating the NGSI-LD data later, when the required *resources* are available.

Relationships can be mapped at three different stages: i) in the oneM2M system before the mapping process; ii) in the SMD during the mapping process; and iii) in the NGSI-LD data after the translation process. At the first and third stage, there are many possibilities where the process can be handled by different services and will be application dependent. Therefore, we limit the scope of this research to the mapping stage.

Considering the second stage, there can be two possibilities for mapping the *Relationships*: the *Relationships* are explicitly defined in the *resource*s; the *Relationships* are not defined in the *resource*s but are explicitly defined in the mapping. In the former case, the mapping will be defined in similar way to the one defined for *Properties* in section IV.A, where the *Relationships* will be mapped using *ngsi-ld:hasRelationship* and the target will be mapped using the mapping ontology. For the latter case, either Semantic Web Rule Language (SWRL) [19], or SPARQL Protocol and RDF Query Language (SPARQL) [20] can be utilized.

Some information is required based on which SWRL or SPARQL can be utilized to map the *Relationship*. Consider a scenario where a *Relationship* "hasParkingSpot", has to be generated between the *Entities* "ParkingLot" and "ParkingSpot". To identify the *Relationship*, two pieces of information can be helpful: local coordinates defined in *resources* for "ParkingLot" and "ParkingSpot"; and the key "parkingLotRef" (defined in the *resource* representing "ParkingSpot"), whose value specifies the id of the *resource* representing "ParkingLot". After mapping the two *Entities*, the SWRL rule or SPARQL query can be defined which can compare the local coordinates of both *Entities* as well as compare the value of "parkingLotRef" with the "annotation:hasEntityId" of the "ParkingLot" *Entity* instance. In the case of SWRL, the resultant *Relationship* will be defined using "ngsi-ld:hasRelationship" (and some other RDF concepts if required) in the consequent of the rule. Upon performing the inference (using an inference engine such as

Pellet [21], HermiT [22], etc.), the appropriate *Relationship* will be generated. However, the utilization of SWRL requires the OWL representation of NGSI-LD meta-model as well as Mapping Ontology, which is straightforward to define following the Ontology defined in NGSI-LD specification. One of the limitations of SWRL-based mapping is that it is highly dependent on the Inference engine: sending a lot of mapping instances to the Inference engine may consume a lot of time resulting in poor performance.

The above limitation can be resolved by using SPARQL, which is flexible as it supports all the CRUD operations as well as some complex operations. In case of the SPARQL query, the antecedent part of SWRL will be defined as the WHERE and FILTER clause, and the consequent will be defined as the INSERT clause. However, SPARQL queries cannot be formally stored and executed using an inference engine like SWRL. They can be stored as a string literal of an RDF property or by utilizing Shapes Constrained Language (SHACL) [23]. Otherwise, a triple database (TDB) enabled with the SPARQL engine is required to store the mapping.

V. USE CASE BASED ON PARKING DATA

Consider a scenario where the parking system utilizes oneM2M-based architecture for managing the parking-related data, which needs to be annotated into NGSI-LD-based LPG for different services such as recommending nearby available parking lots or parking spaces. Different types of *resources* need to be created such as parking lot, parking spot, parking lot congestion, parking floor congestion, etc.

```
{"m2m:cin": {
     ...
     "con": {
         "name": "spot_001",
         "Datelog": { "creation": "2018-11-15T20:10:00,000+09:00" },
         "category": [ "forDisabled" ],
         "loc": {
             "type": "PointSystem",
             "PointSystem": { "Lat" : -2.558246945 ,
                              "Long": 50.378546923 }
         },
         "Dimensions": { "width": "2.5",
                         "Length": "5.1",
                         "unit": "m" }
         "availability": {
            "value": "free",
            "observation": "2018-11-15T20:09:55,000+09:00",
         }
     },
  ...
}}
```

Fig.3. Parking Spot Data in CIN

Fig. 3. shows sample data for a parking spot in the Republic of Korea. Here "…" in the third and third last row represents the data that is not relevant to be shown for annotation. This data usually involves the mandatory or optional *resource* attributes based on oneM2M specifications.

Fig. 4 shows the respective mapping of the parking spot data to a single *Entity*. Most of the RDF resources represent the key or id of the elements in NGSI-LD such as "parking:name", "parking:unit", etc. They act as the value of "targetKey" in the mapping. The URIs of these RDF resources are defined based on the @context for the parking domain. This @context has to be available to the mapping and the translation module in advance. The dot "." representation scheme for *resources* can be observed in the range values of properties "sourceKey" and "sourceValue". At the beginning of each value, "/la" denotes the latest CIN *resource* content in the case of multiple CIN *resources*. A part of the value for "hasEntityId", which is enclosed in "{}", shows the address to be referenced, whereas, the value outside "{}" indicates the static value to be added as the prefix.

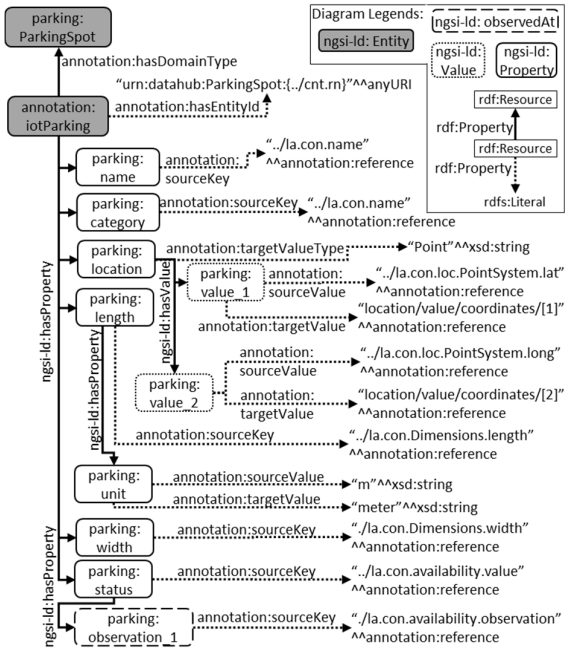

Fig. 4. RDF-based Mapping Representation in SMD. The diagram legends show the rdf:type of the RDF resource and properties.

In Fig. 4, the mapping related to "parking:location" ("loc" in Fig. 3) is an example of complex value mapping, where two "ngsi-ld:Value" instances map the respective longitude and latitude information. The value format of "targetValue" differs from that of the *source*, as the longitude and latitude values are mapped as first and second elements in the array respectively, represented by an index enclosed in "[]". In addition, the property "targetValueType" has been used to explicitly define the *Value* type used in NGSI-LD, which is not determined in Fig. 3. The mapping related to "parking:unit", defined in Fig. 4, shows another usage of "targetValue" based on simple conditioning. The translation system checks if the "sourceValue" is "m", the type in NGSI-LD will be defined as "meter", through interpreting the "targetValue". For multiple conditions, instances of type "ngsi-ld:Value" can be defined, linked together with "parking:unit" using "ngsi-ld:hasValue", where each instance will represent a single condition. The mapping related to "observation", defined in Fig. 4 shows the usage of predefined *Property* "observedAt".

The final NGSI-LD Entity representation can be seen in Fig. 5. Here the value of "id" (which has been mapped using the literal value of "hasEntityId" (see Fig. 4), has "yatap_540" as its postfix, which has been retrieved following the reference "{../cnt.rn}" in the mapping, specifying the parent CNT *resource* attribute.

The current mapping protocol has some limitations regarding *Value* mapping as it cannot provide complex value objects to the full extent. For instance, the time stamp for the *Property* can be mapped if the *resource data* has defined the time stamp value using the XSD data type. It will not support any other custom non-standard time stamp. Support for such complex values and other JSON object structures is the scope of future study.

```
{ ...
   "id": "urn:datahub:ParkingSpot:yatap_540",
   "type": "ParkingSpot",
   "name": { "type": "Property",
             "value": "spot_001" },
   "location": {
      "type": "GeoProperty",
      "value": {
                 "type": "Point",
                 "coordinates": [127.1293735,37.4114423]
      }
   },
   "category": { "type": "Property",
                 "value": ["forDisabled"] },
   "width": { "type": "Property",
              "value": 2.5 },
   "length": {
        "type": "Property",
        "value": 5.1,
        "unit": { "type": "Property",
                  "value": "meter" }
   },
   "status": {
      "type": "Property",
      "value": "free",
      "observedAt": "2018-11-15T20:09:55,000+09:00"
   }
}
```


Fig. 5. Annotated Parking Spot data in NGSI-LD

VI. CONCLUSION

In this paper, we proposed a novel approach to utilize RDF for annotating oneM2M resources into NGSI-LD. The mapping protocol has been applied to the smart parking use case. The mapping can provide an interpretation of value at some level of complexity. The value-to-value mapping with different standards still has limitations and requires manual work. However, this issue can be resolved using reasoning such as using SWRL. Linking the NGSI-LD Entities is also an important aspect of this research whose implementation feasibility is yet to be explored. The resolution of these challenges will improve the mapping capabilities, which is also the future work of this study.

ACKNOWLEDGMENT

This work was supported by Korea Environment Industry & Technology Institute (KEITI) through Intelligent Management Program for Urban Water Resources Program, funded by Korea Ministry of Environment (MOE) (RE201903069).

REFERENCES

[1] M. Yahya, J. G. Breslin, and M. I. Ali, "Semantic Web and Knowledge Graphs for Industry 4.0," *Appl. Sci*., vol. 11, no. 11, p. 5110, May 2021.

[2] G. Buchgeher, D. Gabauer, J. Martinez-Gil, and L. Ehrlinger, "Knowledge Graphs in Manufacturing and Production: A Systematic Literature Review," *IEEE Access,* vol. 9, pp. 55537–55554, 2021.

[3] oneM2M, "Semantic Support", oneM2M, Rep. TS-0034-V3.0.1, 2019.

[4] ETSI, "Context Information Management (CIM); Information Model (MOD0)", ETSI, Rep. ETSI GS CIM 006 V1.1.1, Jul. 2019.

[5] "AWS IoT Things Graph.", AWS IoT Things Graph. https://aws.amazon.com/iot-things-graph/?nc2=h_ql_prod_it_tg%0Ahttps://aws.amazon.com/iot-things-graph/. (accessed Apr. 21, 2022).

[6] A. Souza et al., "A geographic-layered data middleware for smart cities," in *Proc. WebMedia '18: Proc. of the 24th Brazilian Symp. on Multimedia and the Web*., Oct. 2018, pp. 411–414.

[7] B. Rocha, E. Cavalcante, T. Batista, and J. Silva, "A Linked Data-Based Semantic Information Model for Smart Cities," in *2019 IX Brazilian Symp. on Comput. Syst. Eng. (SBESC)*, Nov. 2019, pp. 1-8.

[8] *R2RML: RDB to RDF Mapping Language*, W3C Standard TR/r2ml/, Sep 2012. [Online]. Available: https://www.w3.org/TR/r2rml/. [Accessed: 21-Apr-2022].

[9] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. V. D. Walle, "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data," presented at the Linked Data on the Web (LDOW2014) - Workshop at WWW2014, Seoul, Korea, Apr. 8, 2014.

[10] R. Reda, F. Piccinini, and A. Carbonaro, "Towards consistent data representation in the IoT healthcare landscape," in *DH '18: Proc. of the 2018 Int. Conf. on Digit. Health*., Apr. 2018, pp. 5–10.

[11] P. Gonzalez-Gil, J. A. Martinez, and A. F. Skarmeta, "Lightweight Data-Security Ontology for IoT," *Sensors*, vol. 20, no. 3, p. 801, Feb. 2020, doi: https://doi.org/10.3390/s20030801.

[12] A. F. Skarmeta et al., "IoTCrawler: Browsing the internet of things," in *2018 Glob. Internet Things Summit (GIoTS)*, Nov. 2018, pp. 1-6, doi: 10.1109/GIOTS.2018.8534528.

[13] M. Bauer, "Morphing Mediation Gateway with Management and Configuration Functions R2," Rep. D2.2, ver. 1.0, Aug. 21, 2017.

[14] Y. Saleem, M. Bauer, and S. Jeong, "Semantic Interoperability Components R2," Rep D2.5, ver 1.0, Nov. 17, 2017.

[15] M. Bauer, "IoT Virtualization with ML-based Information Extraction" in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 915-920.

[16] J. G. Almeida, J. Silva, T. Batista, and E. Cavalcante, "A linked data-based service for integrating heterogeneous data sources in smart cities," in *Proc. of the 22nd Int. Conf. on Enterprise Inf. Syst. (ICEIS 2020)*, 2020, pp. 205–212.

[17] A. Detti et al., "System Architecture - First Release," Rep. D2.2, March. 31, 2019.

[18] oneM2M, "Functional Architecture," oneM2M, Rep. TS-0001-V3.27.0, March. 03, 2022.

[19] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Standard SUBM-SWRL-20040521/, May 21, 2004. [Online]. Available: https://www.w3.org/Submission/SWRL/. [Accessed: 21-Apr-2022].

[20] *SPARQL 1.1 Query Language*, W3C Standard TR/sparql11-query/, March 21, 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/. [Accessed: 21-Apr-2022].

[21] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz, "Pellet: A Practical OWL-DL Reasoner," *J. of Web Semantics*, vol. 5, no. 2, pp. 51-53, 2007.

[22] B. Glimm, I. Horrocks, B. Motik, and G. Stoilos, "HermiT: An OWL 2 Reasoner," J. of Automated Reasoning, vol. 53, no. 3, pp. 245–269, May 2014.

[23] *Shapes Constraint Language (SHACL)*, W3C Standard TR/shacl/, Jul. 20, 2017. [Online]. Available: https://www.w3.org/TR/shacl/. [Accessed: 21-Apr-2022].