ETSI GR CIM 020 V1.1.1 (2022-12)

Context Information Management (CIM); NGSI-LD; Guidelines for the deployment of Smart City and Communities data platforms

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
DGR/CIM-0020

Keywords
interaction, interoperability, NGSI-LD, platforms, 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

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

Modal verbs terminology

In the present document "**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.

Executive summary

The present document presents ideas and state-of-the-art solutions, based on the NGSI-LD eco-system (specifically the NGSI-LD API in ETSI GS CIM 009 [i.2], information model in ETSI GS CIM 006 [i.3] and security and privacy in ETSI GR CIM 007 [i.4]), in the area of smart cities, i.e. the area of software platforms which, based on real-time data flows, make it possible to better manage cities and communities.

Introduction

A smart city platform helps to systematically manage and utilize various data generated in the city. It may improve the quality of life of citizens by creating new businesses and services and continuously developing the city.

Concepts like "federation", "interoperability", and "security" should be considered for smart city services. Through an NGSI-LD based smart city platform, the various data generated in the city are systematically managed, and through this, the digital industry ecosystem can be activated by creating new services and business models, or by creating added value to the existing ones.

1 Scope

The present document presents ideas and state-of-the-art solutions in the area of smart cities, that is in the area of software platforms which, based on (near) real-time data flows, make it possible to better manage cities and communities.

Specifically, the solutions are centred on the NGSI-LD API [i.2] and information model [i.3], as a means to achieve cross-domain interoperability among data coming from different sectors and the different (possibly pre-existing) data sources installed in public spaces, utilities and infrastructures.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

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] OASC Minimal Interoperability Mechanisms.

NOTE: Available at https://mims.oascities.org.

[i.2] ETSI GS CIM 009 (V1.6.1): "cross-cutting Context Information Management (CIM); NGSI-LD API".

NOTE: Available at https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.06.01_60/gs_CIM009v010601p.pdf.

[i.3] ETSI GS CIM 006 (V1.1.1): "Context Information Management (CIM); Information Model (MOD0)".

NOTE: Available at https://www.etsi.org/deliver/etsi_gs/CIM/001_099/006/01.01.01_60/gs_CIM006v010101p.pdf.

[i.4] ETSI GR CIM 007 (V1.1.2): "Context Information Management (CIM); Security and Privacy".

NOTE: Available at https://www.etsi.org/deliver/etsi_gr/CIM/001_099/007/01.01.02_60/gr_CIM007v010102p.pdf.

[i.5] IoT Agent Library.

NOTE: Available at https://github.com/telefonicaid/iotagent-node-lib.

[i.6] The Digital Twins Definition Language (DTDL).

NOTE: Available at https://github.com/Azure/opendigitaltwins-dtdl.

[i.7] Digital Twins Definition Language (DTDL) ontology for Smart Cities.

NOTE: Available at https://github.com/Azure/opendigitaltwins-smartcities/blob/main/README.md.

[i.8] FIWARE for Data Spaces.

NOTE: Available at https://www.fiware.org/wp-content/uploads/FF_PositionPaper_FIWARE4DataSpaces.pdf.

[i.9] ETSI White Paper no. 42: "Guidelines for Modelling with NGSI-LD". 1st edition - March 2021.

NOTE: Available at https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_WP_42_NGSI_LD.pdf.

[i.10] The Business API Ecosystem.

NOTE: Available at https://business-api-ecosystem.readthedocs.io/en/latest/.

[i.11] GIS Industry: "How GIS Supports the Planning and Development of Smart Cities". Sangeeta Deogawanka. February 8, 2016.

NOTE: Available at https://www.gislounge.com/how-gis-supports-the-planning-and-development-of-smart-cities.

[i.12] INSPIRE Directive: "Infrastructure for Spatial Information in Europe. D2.5: Generic Conceptual Model, Version 3.4rc3".

NOTE: Available at https://inspire.ec.europa.eu/documents/Data_Specifications/D2.5_v3.4rc3.pdf.

[i.13] Smart Data Models Program.

NOTE: Available at https://smartdatamodels.org/index.php/about/.

[i.14] QGIS Topology Introduction.

NOTE: Available at https://docs.qgis.org/3.22/en/docs/gentle_gis_introduction/topology.html.

[i.15] W3C Recommendation 04 February 2020: "Data Catalog Vocabulary (DCAT) - Version 2".

NOTE: Available at https://www.w3.org/TR/vocab-dcat-2/#Class:Resource.

[i.16] IETF RFC 8428: "Sensor Measurement Lists (SenML)", August 2018.

NOTE: Available at https://www.rfc-editor.org/rfc/rfc8428.

[i.17] IETF RFC 7946: "The GeoJSON Format", August 2016.

NOTE: Available at https://www.rfc-editor.org/rfc/rfc7946.

[i.18] TopoJSON API Specification.

NOTE: Available at https://github.com/topojson/topojson/blob/master/README.md.

[i.19] Antti Poikola, Kai Kuikkaniemi, Harri Honko: "A Nordic Model for human-centered personal data management and processing". ISBN: 978-952-243-455-5.

NOTE: Available at https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/78439/MyData-nordic-model.pdf.

[i.20] SAREF: Smart Applications REFerence Ontology, and extensions: Official ETSI portal for SAREF.

NOTE: Available at https://saref.etsi.org/.

[i.21] OGC API Overview.

NOTE: Available at https://www.ogc.org/standards/ogcapi-features.

[i.22] OGC Open Geospatial Consortium.

NOTE: Available at https://ogcapi.ogc.org.

[i.23] W3C Semantic Web Standards.

NOTE: Available at https://www.w3.org/standards/semanticweb/.

[i.24] DBPedia LodLive.

NOTE: Available at https://github.com/LodLive/LodLive.

[i.25] OGC City Geography Markup Language (CityGML) 3.0 Conceptual Model Users Guide, 13th September 2021.

NOTE: Available at https://docs.ogc.org/guides/20-066.html.

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

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

3D	Three Dimensional
AI	Artificial Intelligence
API	Application Programming Interface
ArcGIS	Graphical Information Software

NOTE From the organization ESRI, see https://www.esri.com.

CBAC	Context-Based Access Control
CSV	Comma Separated Value
DCAT	Data CATalogue vocabulary
DTDL	Digital Twins Definition Language
EDM	Europeana Data Model

NOTE: An initiative funded since 2005 by the EU.

FTP	File Transfer Protocol
GIS	Graphical Information System
GML	Geography Markup Language
HTTP	HperText Transfer Protocol
ICT	Information and Communications Technology
ID	IDentifier
IoT	Internet of Things
IUDX	Indian Urban Data Exchange
JSON	Java Script Notation Object
LD	Linked Data
LOD	Linked Open Data
MIM	Minimal Interoperability Mechanism(s)
MQTT	Message Queue Telemetry Transport
NGSI	Next Generation Service Interface
OASC	Open and Agile Smart Cities
OGC	Open Geospatial Consortium
OKG	Open Knowledge Graph
OWL	W3C Web Ontology Language
PDP	Policy Decision Point
PEP	Policy Enforcement Point
RBAC	Role-Based Access Control

RDBMS	Relational DataBase Management System
RDF	Resource Description Format
REST	REpresentational State Transfer
senML	Sensor Measurement Lists
SKOS	Simple Knowledge Organization System
SMS	Short Message Service
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language,
SWS	Semantic Web Standards
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WCS	OGC Web Coverage Service
WFS	OGC Web Feature Service
WMS	OGC Web Map Service
WPS	OGC Web Processing Service

4 Basics for building a smart city platform

4.0 Introduction

To implement a data-based smart city, a software platform that can systematically manage and utilize the various kinds of data generated in the city is required. Smart city platforms process, store, and convert data collected by the city, and provide it to city services that extract useful information out of data, for example by carrying out predictive analysis based on AI. In this way, the city can provide the building blocks for various types of applications according to the emergence of information out of real-time data such as car/motor traffic volume or other data collected by new technologies. For this purpose, it is necessary to ensure a continuous data flow to the platform, and therefore data collected from various systems and domains would benefit from inter-linking.

Towards this goal, a smart city platform based on the NGSI-LD ecosystem would ensure data interoperability by using and exploiting the NGSI-LD [i.2] API and its associated, cross-domain data model [i.3] to store and deliver data.

4.1 Objectives of implementing a smart city platform

The purpose of the smart city platform is to manage urban data that provides a data-based collaboration, analysis, and decision-making environment, by establishing a real-time linkage of vast amounts of information generated by infrastructure, administration, and civil communities that make up the urban environment through the horizontal/vertical convergence of smart city technologies.

High-level objectives are summarized as follows:

- Establish an urban management model that supports the collective acquisition and sharing of information about various smart city infrastructures and urban conditions required by industry-specific services.
- Create an urban data industry ecosystem for integrated storage, analysis, and utilization of large amounts of data in collaboration with cloud infrastructures.
- Support the interoperability between different domains so that added value emerges.
- Collect city data from heterogeneous data sources such as legacy systems, IoT platforms, and GIS, in order to process, analyse, and distribute data to city services and other external systems.
- Link the data via semantic web technologies for better data usage, cost minimization for the services, and improved usability of data from various domains.

The expected features of a smart city platform are:

- Data ingestion & integration from various domains
- Cloud-based document and multimedia storages
- Assets management
- Data analysis and monitoring
- Security & Access control for users

4.2 NGSI-LD-based smart city platform architecture

The smart city platform needs to implement technologies to collect data from a variety of external platforms and convert and adapt it to already existing data models for smart cities, e.g. Smart Data Models [i.13].

Using data models which are compatible with the NGSI-LD cross-domain information model allows a high degree of data inter-linking when an NGSI-LD broker is used as the central piece of this architecture.

Various external IoT data sources (e.g. based on oneM2M), Geographical Information Systems, legacy systems, as well as other smart city platforms can be considered as input to the smart city platform, which is then implemented as a cloud-based system that collects, processes, and stores information about the various smart city infrastructures and urban situations.

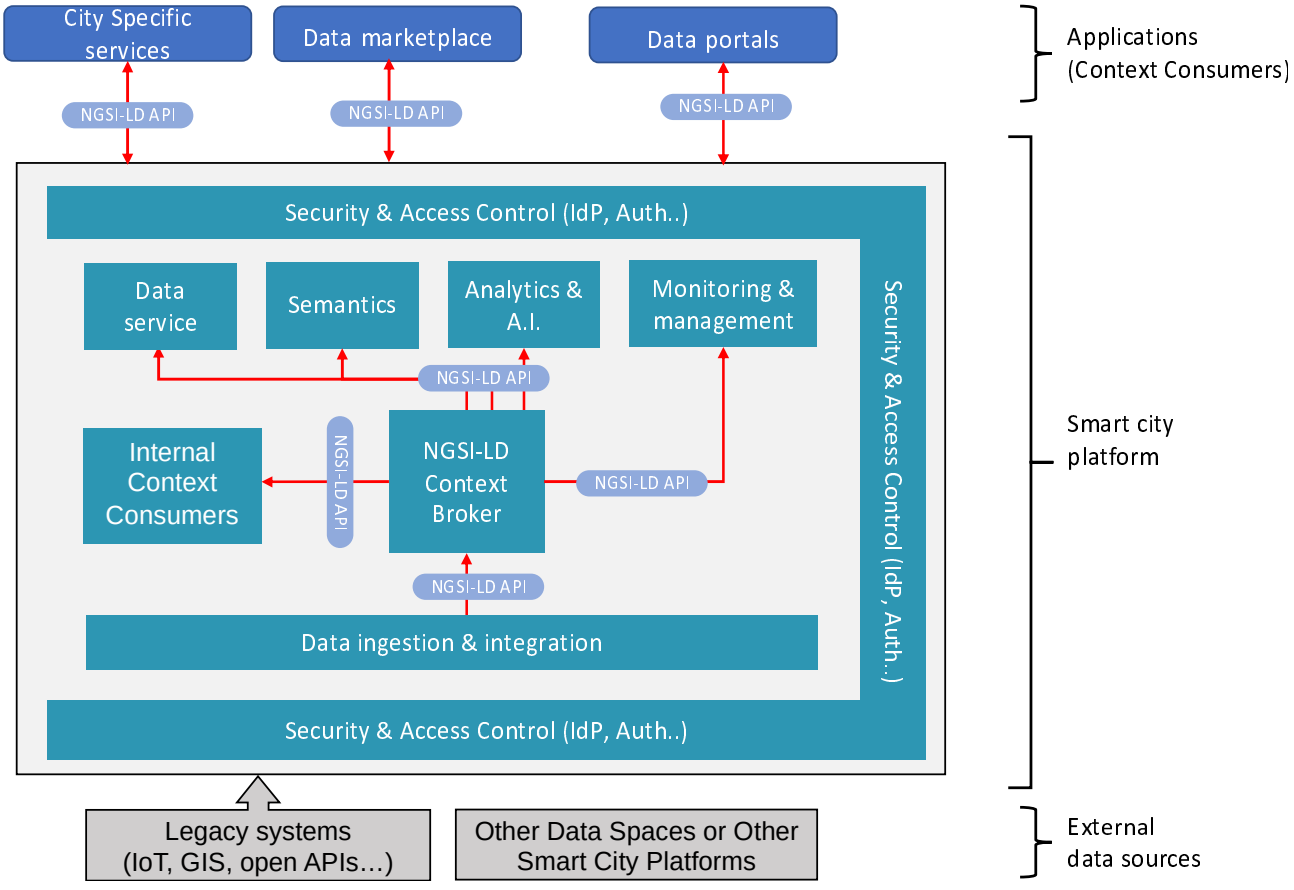

Figure 4.2-1: The structure of an NGSI-LD-based smart city platform

In Figure 4.2-1 the overall architecture of a typical smart city platform is depicted. The platform itself is represented by the middle box, which collects data from external sources and peer platforms (at the bottom, grey boxes), and feeds information to external applications on top (blue boxes).

Logical functions, the green boxes, which are part of the structure of the smart city platform are:

- Data ingestion & integration: collects, transform, and load heterogeneous data from several types of systems such as IoT platforms, RDBMS, and open APIs. When data needs to be transformed to fit the NGSI-LD information model used by the smart city platform, this function could be used as an adapter to load the data.
- NGSI-LD Context Broker: ensuring data interoperability by applying NGSI-LD API and information model to data storage and provision.
- Semantics: supports the construction and utilization of semantic data of the smart city platform. This function converts city data to semantic (RDF triples) data, based on smart city ontologies, to support application services such as Linked Open Data (LOD). Since new knowledge could be extracted through semantic technologies such as semantic mashup and semantic reasoning, it has a bidirectional connection with the NGSI-LD context broker. See LOD, described in clause 11.
- Analytics & Artificial Intelligence (AI): supports the development of a convergence analysis/prediction service of city data. This function analyses the data stored in the smart city platform and creates a machine learning model, and execution of the generated model enables the generation of prediction information. See Digital Twins for smart cities, described in clause 6.
- Monitoring & management: provides real-time integrated monitoring and system operation management such as public/private data and cloud management, devices and database management in the smart city platform. See clause 5.
- Security & Access Control: provides authentication for smart city platform users and applications, access control policy management, and access control token management functions. See Data Sovereignty, Trust and Privacy, described in clause 10.

4.3 Smart city and communities

Collaboration is the key component in a smart city with technology and citizen involvement. Because each smart city platform is different in many ways, exchanging knowledge or data and experience is necessary for smart and sustainable cities and communities. Smart cities could be smarter together and build on their strengths through sharing and reuse of resources and knowledge with communities. The city context broker could have a federated/distributed structure that is shared with other brokers, allowing distributed collection according to domains or data types such as IoT, infrastructure, and document, which it then provides the smart city services.

Figure 4.3-1 presents an example of federated context brokers. It is based on one city context broker that is federated with a set of community context brokers. The community context brokers may share (some or) all their data with the city context broker. This is according to the hierarchical relation between cities and communities. When communities are physically sharing some resources, such as a river or roads, it is possible to create s secure channel between community context brokers to partially share common data. A community context broker can, in turn, be built on a federated architecture where the sub-context broker is defined according to the dedicated domain. The federated and distributed architecture of the NGSI-LD context brokers fully supports this desired hierarchy relation between cities and communities.

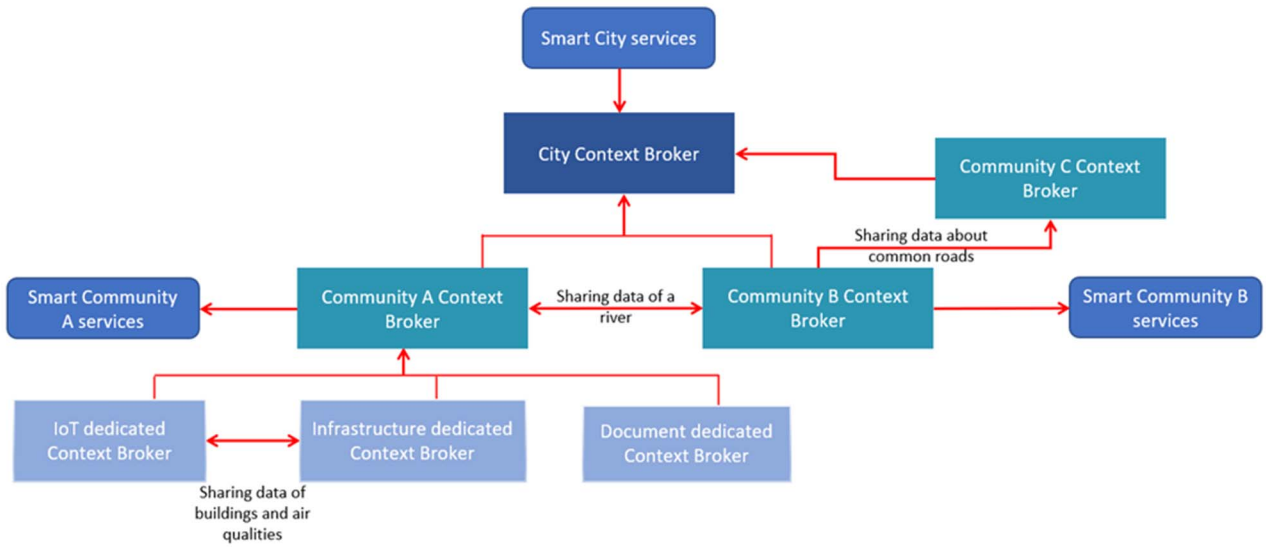

Figure 4.3-1: The distributed/federated structure of smart city and communities

A more complex approach to federation is gaining traction in the past few years, implemented in practice using the concept of a Data Space composed of many building blocks. The concept of federation needs to be augmented with Trust and Identity Providers, in order to form an effective, shared Data Space, as explained in clause 7.

4.4 Public lights in smart city example use case

This clause describes an example of the lifecycle of a public tender implemented and managed through an NGSI-LD context broker in a smart city. It details the main actions requested for a public lighting tender. Figure 4.4-1 presents an overview of how it would be possible to manage a public tender in an NGSI-LD-based city platform. The tender's life cycle progress, steps, and its projection on NGSI-LD are detailed in the following.

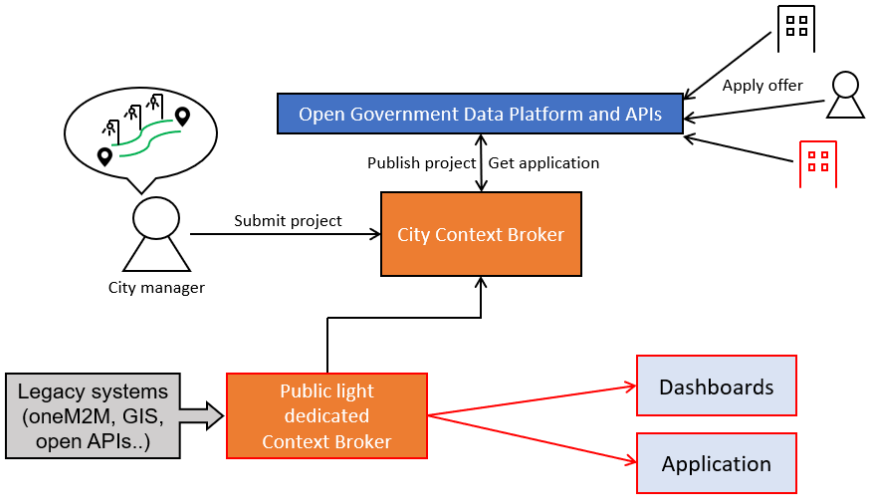

Figure 4.4-1: Scenario overview for managing a public tender in a NGSI-LD-based city platform

- Submitting the public tender on the city platform:

  After agreeing on the public tender internally, the city will start by publishing it on its platform. A public tender is usually detailed by documents, objects (videos, images, etc.) and limited by deadlines. For multimedia and object storage using context brokers refer to clause 5.3. Tender metadata (short description, budget, deadline, requirement, etc.) and relations (responsible, related cities, etc.) may be modelled as NGSI-LD Properties and Relationships.

- Publishing the public tender on official government open data:

  To make the tender public, its description and related objects should be published in official government open data platforms. Official government APIs will have the access to the public tender details on the city platform. This requires the adaptation of these APIs or the use of external services that publish the tender from the city context broker to the open government data platforms. This external service for an open government data platform could be one of the data portals in Figure 4.2-1.

- Applying for the public tender:

  Using the official government APIs, public or private persons may apply to this tender with an offer that meets the needs of this public tender in terms of supplies, services, and works. Offer details (description and objects) and organizations need to be injected on the city platform and related to the concerned tender. The offer information (applier, method, goal, etc.) may be modelled as NGSI-LD Properties and Relationships so that every offer could be managed via the NGSI-LD context broker.

- Project management and progress monitoring:

  Once the public tender deadline is reached, the city will select one of the available offers to be the winner of the tender. Project deadlines, technical aspects, milestone monitoring, and budget progress may be monitored by the NGSI-LD compliant city platform by injecting data. This kind of process could be a practical example of monitoring in the smart city platform mentioned in clause 4.2.

- Project implementation:

  The project may require a dedicated context broker that collects data related to public lights in the smart city through oneM2M, open APIs, and other legacy systems connected to the city context broker with distributed and/or federated structures with the city data platform. An application layer maybe then built on top of the dedicated context broker for executing smart data-based applications. To develop, for instance, a dashboard or an application meeting the requirements of the domain specific project, access control of each dedicated context broker over the city context broker should be managed, as outlined in clause 10.

The public lights in the smart city use case is an example of a scenario that could currently be implemented with NGSI-LD-based technologies introduced in the present document. The distributed structure referred to in clause 4.3 supports domain specific dedicated context brokers that can be given access to the developer, and the object storage referred to in clause 5.3 makes it possible to manage not only the text of the project offer of the tender, but also various multimedia materials.

5 Data sources for smart cities

5.0 Introduction

External data sources are crucial for smart cities and are used for injecting data. These data sources are considered as the location where pieces of data are sourced. Technically, data sources can be simple systems or even complex platforms, and using a variety of technologies, such as:

- Other NGSI-LD context brokers that are federated with the one(s) running the smart city platform.
- Message brokers (e.g. an MQTT broker), for attaching to raw streams of data.
- Open data platforms such as the ones hosting open government data.
- REST APIs-based Internet services such as weather or traffic APIs.
- IoT platforms such as complex oneM2M-based deployments.
- File content on a server such as text, CSV, spreadsheets, SQL dumps, etc.
- Multimedia objects on a server such as images and videos (could also be real-time).

The goal, when integrating a data source within a smart city platform, is to be able to interface the platform with the world of connected objects and data. Incoming data from external data sources is characterized by:

- An upstream data flow
- Heterogenous data formats
- Heterogenous data sources
- Highly variable data sending cycles/rates
- A potentially large number of sensors in the case of IoT platforms

5.1 Integrating data sources in an NGSI-LD platform

In this clause, the discussion assumes that the city platform is NGSI-LD compliant (thus, with an NGSI-LD context broker at the centre of its architecture, as depicted in Figure 4.2-1), and possible ways of integrating it with external data sources are depicted. Several modes of data source integration can be considered by the city, each option having different technical and business impacts. Figure 5.1-1 details these modes of integration:

Option A: The data source is deployed by a third party (municipality, etc.) or maintained by the same city, by following the principles of federated context brokers presented in clause 4.2. External applications, in this case, are expected to communicate with the city platform via the NGSI-LD API.

Option B: The data source, in this case, is expected to be non-compliant to the NGSI-LD system. The data source could be a oneM2M based IoT platform, or a stream of Linked Open Data for example (see clause 11). The city's platform then should provide aggregators and interworking proxies capable of adapting to the different protocols, data models and security modes used by the data source platform to NGSI-LD compliant format.

Option C: Equivalent to option B with the application layer developed by a third party and assumed to be non-compliant with NGSI-LD. The application backend will be in charge of consuming data from the city data platform and providing it to the application.

Option D: The entire chain from the data source to the application is not NGSI-LD compliant in this case. The data source could be an open government data portal, or a stream of Linked Open Data for example. Data sources inject data directly to the backend applications, and data is then, in turn, injected into the city's NGSI-LD context broker by the application backend.

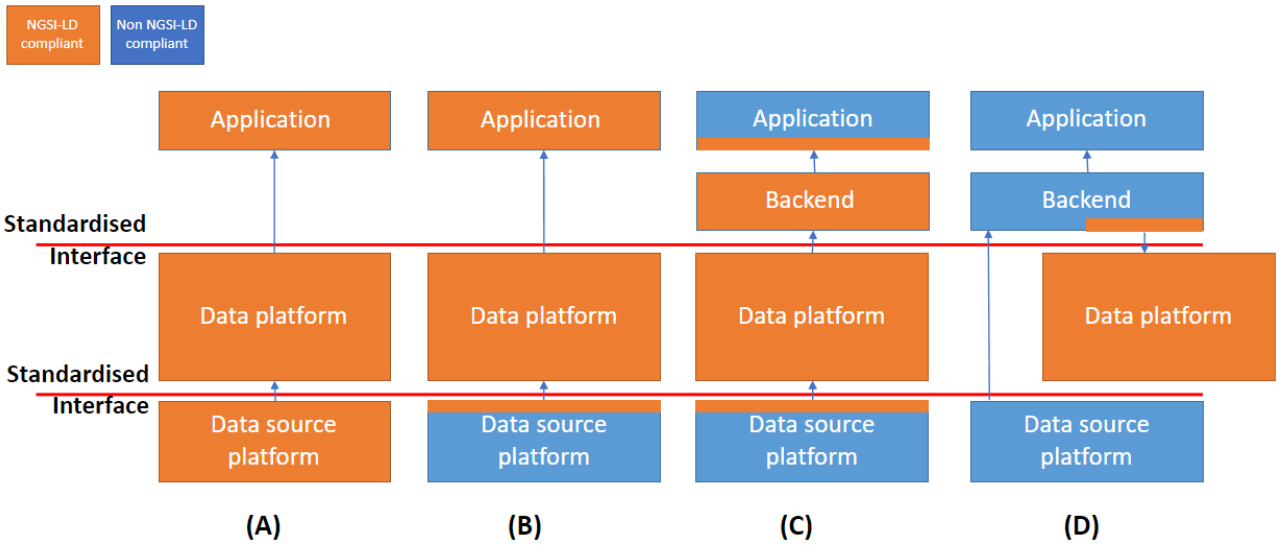

Figure 5.1-1: Integrating external data source platforms with an NGSI-LD compliant data platform

5.2 Non-NGSI-LD-compliant data source management

This clause presents one of the possible ways for integration of non NGSI-LD compliant data sources with the city platform.

Note that the following clauses should be seen as guidelines, and they are showing only one of the possible ways for integrating of data source platforms with NGSI-LD compliant ones.

In this clause, an example of integrating an external data source with NGSI-LD data platform is presented. The proposed approach follows partially the IoT Agent node library [i.5] approach, augmented with a full NGSI-LD-based provisioning. Specifically, based on the features of the NGSI-LD API, the context broker could play the role of managing data sources by means of creating correspondent entities. To consume data from data source, an external "data source management layer" is needed. The data source management layer is in charge to register on the data platform and then to consume and produce NGSI-LD compliant data. Figure 5.2-1 details the main architecture of integrating a data source platform and management layer with the NGSI-LD compliant platform. The steps toward this integration are as follows:

- Register the data source management layer on the city platform by creating its correspondent entity.
- Create data source and mapping entities on the city platform.
- Extract and consume data about the created data source and mapping entities by the data source management layer.
- Subscribe or connect on the data source platform by the data source management layer.
- Consume raw data from source platform.
- Format this raw data to follow the NGSI-LD structure with the help of details from the correspondent mapping entity.
- Query the city data platform to inject this data.

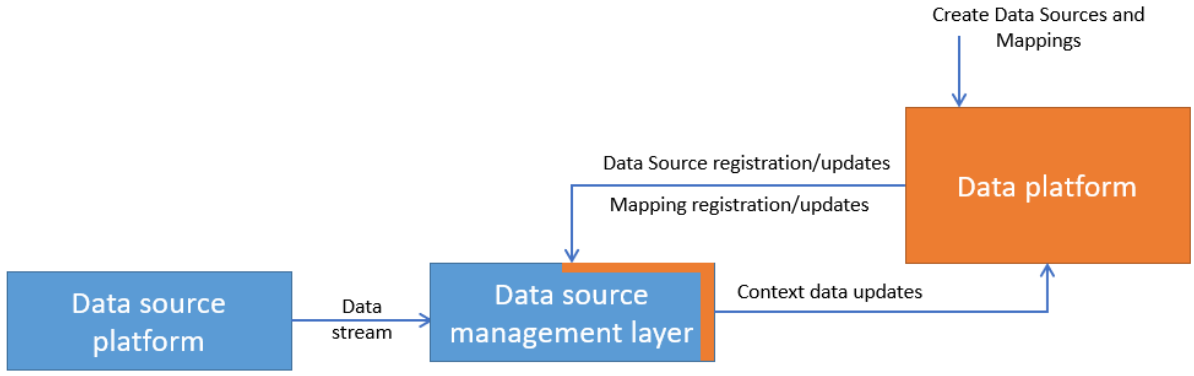

Figure 5.2-1: Example of integrating Data Source platform and Management Layer with NGSI-LD compliant platform

On the city data platform, it is possible to create, update and delete data sources and their mappings using NGSI-LD specific data models. A data source and its mappings may be represented as NGSI-LD entities, that is the Data Source entity and the Mapping entity:

- The Data Source entity will contain details about the data source: that could be an IoT platform such as oneM2M, a Lora Server, an FTP server, set of multimedia documents, or any generic source of data. The Data Source entity may for example use the "dcat:Resource" class data model from the DCAT ontology [i.15]. Additionally, the Data Source entity has to contain properties about all parameters needed to establish a secured connection to the source.

EXAMPLE: MQTT broker as a Data source. For this kind of data source, the Data Platform requires the URL of the MQTT broker, topics to subscribe and credentials. URL, topics, credentials, and data source type (MQTT) are handled as properties of this data source entity.

- The Mapping entity will relate the data source to its context entity. This type of entity will model the destination of each incoming data. Mapping entities will play the role of mapping each incoming data to its corresponding NGSI-LD context.

To understand this architecture, in the following an example of a Weather Service data source provided over an MQTT broker using the senML [i.16] JSON format is described. Registering this data source in the context broker is done by creating an entity of type Data Source. An example of such a Data Source entity is depicted in Figure 5.2-2.

```
{
    "id": "urn:ngsi-ld:DataSource:01",
    "type": "DataSource",
    "topics": {
        "type": "Property",
        "value": "weather/#"
   },
    "url": {
        "type": "Property",
        "value": "localhost"
    },
    "port": {
        "type": "Property",
        "value": "8383"
    },
    "qos": {
        "type": "Property",
        "value": "01"
   },
    "certificate": {
        "type": "Property",
        "value": "xyz"
    },
"@context": [
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.5.jsonld"
    ]
}
```

Figure 5.2-2: Data source Entity example

Suppose that the following senML message is sent over an MQTT broker:

```
[{"bn":"urn:dev:weatherDataSource","bt":1.59196398E9},{"n":"t","v":25},,{"n":"r","v":95}]
```

Figure 5.2-3: Example senML message

It corresponds to a weather data source that provides the temperature of a city with a level of reliability.

The NGSI-LD data model of the Mapping entity and an example instance of a mapping entity are depicted in Figures 5.2-4 and 5.2-5, respectively.

On the Mapping entity:

- The property "deviceId" corresponds to the "bn" of the senML message. This id will be exploited by the management layer to identify the device. (In a real use case, the management layer will consume messages from different sources identified by this "bn").

- The property "attributesToMap" is a multi- instance attribute. Each instance corresponds to the mapping of one of the device parameters:
  - The "value" corresponds to the name of the parameter of the senML message.
  - The sub- property "subAttribute" corresponds to the name of the sub-parameter field.
  - The value of the property named "attribute" is the name of the corresponding property of the mapped entity. In this use case, the entity "urn:ngsi-ld:City:01" contains a property with the same name as the value of the mapped attribute (temperature).
  - The sub- property "attributeDatasetId" contains the value of the datasetId of the corresponding context entity, property and instance.
  - The "hasTargetEntity" relationship relates the mapping instance to its context entity.

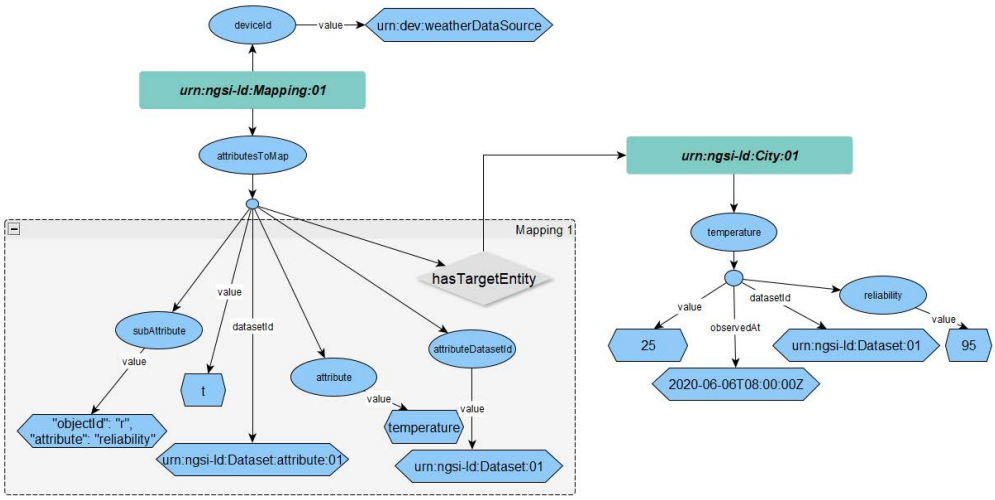

Figure 5.2-4: Graphical presentation of the NGSI-LD data model of the Mapping entity

```
{
"id":"urn:ngsi-ld:Mapping:01",
"type":"Mapping",
"deviceId": {
   "type": "Property",
   "value": "urn:dev:weatherDataSource"},
"attributesToMap": [
      {
         "type": "Property",
         "value": "t",
         "attribute": {
            "type": "Property",
            "value": "temperature"
         },
         "attributeDatasetId": {
            "type": "Property",
            "value": "urn:ngsi-ld:Dataset:01"
         },
         "hasTargetEntity": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:City:01"
         },
         "subAttribute": {
            "type": "Property",
            "value": {
               "objectId": "r"
               "attribute": "reliability"
            }
         },
         "datasetId": "urn:ngsi-ld:Dataset:attribute:01"}
   ],
 "@context": [
https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.5.jsonld
] }
```


Figure 5.2-5: Example of NGSI-LD mapping entity

The data source management layer should handle the status of these type of entities from the data platform, using for example NGSI-LD subscription API (subscription to all entities of type Data Source and Mapping). The communication with the NGSI-LD broker and this layer may be via the subscription mechanism: the data source management layer will subscribe to any changes on entities of type data sources and mappings or HTTP calls, to be notified of the latest status of these entities.

The proposed architecture allows the support of several data source management services at the same time on the same context broker.

The NGSI-LD broker should support authentication and provide entity authorization policies (see clause 5.4) to manipulate entities of types of data source management service, data sources, and mappings.

5.3 Document management and cloud object storage

In a smart city environment, external assets such as videos, pdf documents, csv and text files and their metadata are often produced, and these should be correctly handled by the data platform. Towards this goal, a cloud storage platform may be integrated with the NGSI-LD context broker such that it follows the same principles proposed in clause 5.2, considering that multimedia documents may also be seen as data source. Figure 5.3-1 details the main architecture of integrating a cloud object storage with a NGSI-LD compliant platform.

The data source management layer at this level will have to:

- Get instructions (on objects) from the NGSI-LD data platform, whenever an entity of type "object":
  - is "created" on the data platform, the data source management layer will exploit details of this entity to push the new object to the cloud object storage;
  - is "deleted", the data source management layer will delete the correspondent object on the cloud object storage;
  - is "updated", the data source management layer will update the correspondent object on the cloud object storage.
- Get objects from external data source platform when new objects are created.
- Push or update these objects on the cloud object storage platform, according to the instructions received from the data platform.
- Inform the data platform about the object status on the cloud object storage platform (such as created, updated, etc.).

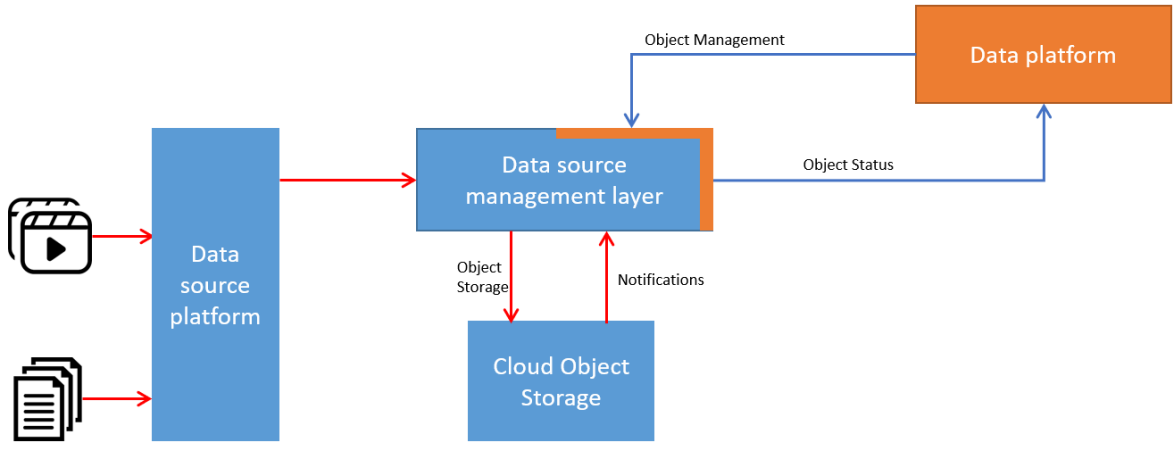

Figure 5.3-1: Example of integrating a cloud object storage platform with NGSI-LD compliant platform

Figure 5.3-2 details an example instance of an entity of type Object. The main attributes of this entity are:

- sourceLocation: contains details of the object which will be stored on the cloud object storage platform.
- cloudLocation: contains the URL of the object on the cloud object storage platform.
- objectStatus: is a temporal property updated by the data source management layer and will contain details about the status of the correspond on the cloud object storage platform.

```
{
    "id":"urn:ngsi-ld:Object:01",
    "type": "Object",
    "sourceLocation": {
       "type": "Property",
       "value": "ftp://user:password@host:port/path"

    },
    "cloudLocation": {
       "type": "Property",
       "value": "\home\user\"
    },
     "objectStatus": {
       "type": "Property",
       "value": "created",
        "observedAt":"2020-06-06T08:00:00Z"
    },
    "@context": [
       "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.5.jsonld"
    ]
 }
```


Figure 5.3-2: Example of an entity of type Object

5.4 Data source authorization over NGSI-LD

One of the important aspects to take into account when injecting data from external sources, is managing authorization for creating, accessing, and modifying entities in the context broker. The main idea of this clause is to provide a model and some examples of policies, which can be useful for managing authorization on NGSI-LD entities such as the Data Source, the Mapping and the Object entities that have been introduced in the previous clauses.

The proposal is to model the authorization management via dedicated special entities of type User and Group, that have NGSI-LD relationships to the generic context entities.

In this way, the security and access control layer presented in Figure 4.2-1, will be in charge to check if users are authorized to create, modify, or update a set of entities, by checking the existence of their corresponding User and Group entities and their rights, within the NGSI-LD context broker itself, rather than involving external proxies. Thus, the following should be seen as a possible solution of implementing the authorization and access control layer directly in the NGSI-LD context broker, potentially making it more efficient (albeit not decoupled) than a completely orthogonal implementation that delegates security to proxy components external to the NGSI-LD context broker. Requirements for security and privacy, described in ETSI GR CIM 007 [i.4], have to be met.

A User entity models a context broker user. A user may be a member of a group using the Relationship *isMemberOf*. A user will inherit policies from the group of which it is a member. A user may have some policies on specific entities using the relationships: *canRead*, *canWrite* and *canAdmin* on the concerned entities. The *canRead* right provides to the user the right to read an entity. The *canWrite* right provides to update an entity. The *canAdmin* right provides to the user the right to administer an entity. The property role - with possible values creator, viewer and admin - defines the role of the user on the context broker. A creator user has the rights to create entities on the context broker, the viewer has only the right to view entities whereas the admin user may have both kinds of rights. Users are admin of their created entities.

```
{
   "id":"urn:ngsi-ld:User:01",
   "type":"User",
   "roles": {
      "type": "Property",
      "value": ["Creator"]
   },
   "isMemberOf": [
      {
         "type": "Relationship",
         "object": "urn:ngsi-ld:Group:01"
      }
   ],
   "@context": [
      "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.5.jsonld"
   ]
}
```

Figure 5.4-1: Example of User entity, where user is member of a Group

A Group entity represents groups. A group is composed of a set of users. A group has the rights of administering entities using the relationships: *canRead*, *canWrite* and *canAdmin* on the concerned entities.

As an example of a policy about registering the data source management layer and data source on the context broker (as seen in clause 5.2), an entity of type user (see above Figure 5.4-1) could have been created in the context broker. This user will create entities of types Data Source and Mapping. When this user creates them, the authorization management layer will update the user entity as follows.

```
{
   "id":"urn:ngsi-ld:User:01",
   "type":"User",
   "roles": {
      "type": "Property",
      "value": ["Creator"]
   },
   "canAdmin": [
      {
         "type": "Relationship",
         "object": "urn:ngsi-ld:DataSource:01",
         "datasetId": "urn:ngsi-ld:Dataset:01"
      },
     {
         "type": "Relationship",
         "object": "urn:ngsi-ld:Mapping:01",
         "datasetId": "urn:ngsi-ld:Dataset:02"
      }
   ],
   "@context": [
      "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.5.jsonld"
   ]
}
```


Figure 5.4-2: Example of updated user entity

The User `urn:ngsi-ld:User:01` will inherit these policies and will have the rights of viewing these entities.

Supposing the group `urn:ngsi-ld:Group:01` is viewer on a set of data source entities, as well, then in Figure 5.4-3 an entity representing the policy of the group is detailed.

```
{
    "id":"urn:ngsi-ld:Group:01",
    "type":"Group",
    "roles": {
       "type": "Property",
       "value": ["Creator"]
    },
    "canRead": [
      {
         "type": "Relationship",
         "object": "urn:ngsi-ld:DataSource:01",
         "datasetId": "urn:ngsi-ld:Dataset:01"
      },
     {
         "type": "Relationship",
         "object": "urn:ngsi-ld:DataSource:02",
         "datasetId": "urn:ngsi-ld:Dataset:02"
      }
    ],
    "@context": [
       "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.5.jsonld"
    ]
}
```

Figure 5.4-3: Example of Group Entity

5.5 Criteria of interest to NGSI-LD

Based on the previous clauses, the main features , some of which still need development, for integrating external data sources with an NGSI-LD compliant platform are listed below:

C1 Secure data source registration: additional mechanisms are needed to securely provision a data source with the necessary security keys and certificates to successfully connect. More general security issues of NGSI-LD are the subject of the security and privacy work item, see ETSI GR CIM 007 [i.4]. The recommendation here is to provide a simple interface to implement group policies for managing data sources over NGSI-LD (see clause 5.4).

C2 Remote management: mechanisms already exist to update object parameters to reflect a new desired state required by the business application, or in response to an external stimulus that prompts modification of the device. This also includes, in the case of actuators, the ability to trigger actions.

C3 Fault management: additional mechanisms are needed for reporting errors occurring during integration of external data sources, application services, etc.

C4 Transforming business data: data received from non-NGSI-LD sources or sent to them needs to be transformed into the appropriate format. It should be possible to manage a set of transformation algorithms through a customizable library.

C5 User management: It is necessary to be able to manage user's access rights according to roles associated with users or user groups. The roles may make it possible to delegate the management of sources or groups of sources to business entities, while allowing the platform administrator to maintain an overall view of the fleet of connected sources.

Additional functionalities, which are traditionally found in data source application platforms but not (yet) in the NGSI-LD API, are listed below. Specification of these functionalities should not compete with the data platform but optimize the general operation (e.g. by reducing the amount of data transferred to the strict minimum). These functionalities include:

C6 Pre-processing: functions for data validation, filtering (e.g. elimination of stationary values), averaging, etc. so as to improve the quality and to optimize the quantity of data sent to the data platform to make it more usable without overloading the databases.

C7 Rules engine: rules, ideally graphically constructed, should allow actions to be triggered (e.g. sending an email or SMS alert) for example when reception of a flow is interrupted (device manager alerts) or on exceeding the preset threshold (business alerts).

C8 Application enablement: simple dashboards should allow presentation of management data or business process results (e.g. evolution of a temperature) on temporal evolution curves, diagrams, or maps.

Table 5.5-1 summarizes how NGSI-LD may support these functionalities.

Table 5.5-1: How NGSI-LD supports criteria of interest for data sources

Criteria	NGSI-LD Supports
C1	As discussed in clause 5.2, modelling the data source management process is possible in NGSI-LD. However, business actions should be delegated for external NGSI-LD compliant services (e.g. the data source management layer).
C2	This is fully supported by NGSI-LD.
C3	There are mainly 3 levels of data validation process. Syntactic data validation, NGSI-LD compliance and semantic+schema. Currently NGSI-LD API returns errors in the case of non-compatible JSON format data and issues with entities and entities parameters (non-existing entity or parameter). However, NGSI-LD does not support constraints or schema-based data validation.
C4	This should be supported by external services, such as IoT agents, that consume raw data and provide NGSI-LD compliant ones.
C5	Not yet supported by NGSI-LD. It may be modelled via NGSI-LD and then delegated to external services.
C6	Aggregation functions are supported by NGSI-LD. However, it is not yet possible to trigger these types of functions automatically. EXAMPLE 1: Store only the average of the last 10 measures. As well as there is no support for aggregation functions on complex JSON objects.
C7	NGSI-LD query language supports rules. However, rules such as "trigger if there is no data for a range of time" are not supported. EXAMPLE 2: If a device updates its data each fixed amount of time, using the API, it is not possible to know if this device has passed the preset fixed amount of time except by checking manually the value of *observedAt* field of the measurement.
C8	NGSI-LD provides features that support temporal evolution of entities that should help to build dashboards with external dashboarding services. The main remaining issue is that the temporal evolution of metadata (such as sub-properties and sub-relationships), as well as the temporal evolution of specific field of complex JSON objects, cannot yet be queried.

6 Digital Twins for smart cities

6.0 Introduction

A Digital Twin is an entity which digitally represents a real-world physical asset (e.g. a bus in a city, a milling machine in a factory) or a concept (e.g. a weather forecast, a product order).

A Digital Twin for smart cities is the way the whole city is represented or managed. There could be sub-Digital Twins (e.g. a zone) in a smart city. A Digital Twin representation of the City gives a real time visibility of various entities and attributes in the city. Holding a digital twin representation of the real-world objects and concepts and describing what is going on in the city requires modelling streets, waste bins and containers, waste trucks, buses, electric vehicle chargers, buildings, events, citizen claims, dog licenses, etc. The different vertical smart solutions deployed in the city (e.g. Air Quality Monitoring, Smart Traffic Management, Smart Parking, Smart Waste Management) are connected to the Context Broker(s), contributing the information that each Broker manages, which is relevant for creating a holistic Context/Digital Twin representation of the whole city, thereby breaking the information silos.

By exploiting the complete Context/Digital Twin representation of the city, a so-called Smart City Governance System (or City Operation Centre) can be developed. Right-time BigData processing tools can be used relying on data coming from multiple sources, extracting more valuable insights for the support of decisions. Similarly, monitoring tools can leverage this holistic Context/Digital Twin representation of the city.

Figure 6.0-1 shows how a Digital Twin of the Smart City is ideally composed of sub-Digital Twins for selected verticals.

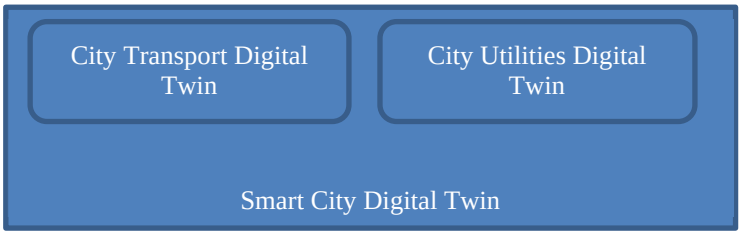

Figure 6.0-1: Digital Twin of the smart city composed of two sub-Digital Twins

6.1 Digital Twin Definition Language

Digital Twins Definition Language (DTDL) [i.6] is an example of a language for describing models and interfaces for digital twins. This language is defined in an open source community and it is based on open W3C standards such as JSON-LD and RDF.

In the case of Smart Cities, a DTDL ontology for Smart Cities is being developed [i.7] that is based on NGSI-LD. A generic NGSI-LD to DTDL Information Model Mapping is under development, as well as NGSI-LD models for Smart Cities mapped in DTDL.

For other domains a detailed analysis is needed about the compatibility between DTDL and NGSI-LD.

6.2 Implementing digital twins with NGSI-LD

6.2.0 Introduction

Digital Twins are key for representing physical spaces in a digital world, but challenges exist in how to organize the data and model the data.

The most common scenario is, there is a physical entity which is not changing, but some of its attributes are changing. For example, a bus is an entity whereby its colour, number of seats and license is static, but its location and speed are changing all the time.

6.2.1 Digital Twin description

Each Digital Twin:

1) is universally identified with a URI (Universal Resource Identifier);

2) belongs to a well-known type (e.g. the Bus type, or the Room type) also universally identified by a URI;

3) is characterized by several attributes which in turn are classified as:

   a) properties holding data (e.g. the "current speed" of a Bus, or "max temperature" in a Room); or

   b) relationships, each holding a URI identifying another Digital Twin entity the given entity is linked to (e.g. the specific Building where a specific Room is located).

Attributes of a Digital Twin may vary, ranging from attributes that are quite static (e.g. the "license plate" of a Bus), to attributes that change very dynamically (e.g. the "speed" or "number of passengers" in a Bus), to attributes which still change but not that often (e.g. the "driver" in a Bus which may change twice a day). Very importantly, the attributes of a Digital Twin are not only limited to observable data but also inferred data. Thus, for example, the Digital Twin of a Street may not only have an attribute "current traffic", which may be automatically measured through sensors or cameras, but an attribute "forecasted traffic in 30 minutes" which might be calculated based on AI algorithms that keep the value of this attribute updated based on current traffic data, other relevant data impacting traffic (e.g. current weather observed and forecasted, schedule of events nearby, etc.) and historical information about traffic in the given street. Therefore, the Digital Twin data representation of the world that is managed is expected to contain all the information needed by smart applications, not only measurable data but also other augmented insights and knowledge acquired over time.

All the Entities in the Digital Twin should be defined to be linked with each other with multiple contexts so that the Digital Twin hierarchy can be handled. The complete data model should be understandable by both human developers and machine applications.

6.2.2 Recommendations to NGSI-LD

The following recommendations would make NGSI-LD more efficient when used to construct Digital Twins:

1) In Smart City Digital Twins, it is essential to have a near real-time snapshot of entities. For example, the activities which are happening on a particular street in the city. To get this snapshot, the data model has to be defined as per the recommendations mentioned at the Information Model work item of NGSI-LD [i.3].

2) It would be valuable to enable a query that efficiently gets the real-time value of an Entity using the value of an attribute of a different Entity, such as a query to "Get all the buses on Main Street at this point in time with speed greater than 30 kmph". Currently, depending on the chosen data model, this type of query may be complicated.

3) It would be valuable to enable creating real-time groups of attributes based on the use cases. Every real-time evolution of a Digital Twin could be mapped to a real-time set of attributes, forming a group. Using attribute groups, only the required attributes could be easily queried. For example, "Show the publicly available information about those people (e.g. who have received a vaccination)".

   One possible way to achieve attribute groups is to use a custom set of @contexts to be hosted at some URLs, that contain all the relevant key values needed to get the value of the required attributes. A single query should be able to get all the data needed. This is further explained in clauses 7.4.3 and 7.4.4.

The above 3 recommendations would improve the Smart Cities Digital Twin performance and adaptability, where information is retrieved based on the use cases supported by the Digital Twin.

7 Smart cities and Data Spaces

7.1 Relationship among Data Spaces, Digital Twins and Context information

A data space can be defined as a decentralised data ecosystem built around commonly agreed building blocks, enabling effective and trusted sharing of data among participants. Any software architecture for Data Spaces gravitates around the management of a Digital Twin data representation of the real world. This Digital Twin data representation is built based on information gathered from different sources, including sensors, cameras, information systems, social networks, end users through mobile devices, etc. It is constantly maintained and accessible in near real-time. "Right-time" is the term often used, reflecting that the interval between the instants of time at which some data is gathered and made accessible is short enough to allow a proper reaction or timely human decision. Applications constantly process and analyse this data (not only current values but also the history generated over time) to automate certain tasks or bring support to smart decisions by end users. The collection of all Digital Twins modelling the real world that is managed is also referred to in the literature as Context, and the data associated with attributes of Digital Twins is also referred to as context information.

7.2 Relevance to Smart Cities

Smart cities are evolving to be platforms which are collecting and providing data from many systems and IoT devices directly. It is not a single monolithic platform. Rather it is a combination of many platforms, which are holding data from many sources. These services can be referred to as Data Spaces.

Smart Cities need to connect to other Data Spaces to share the data across many other systems relevant to it and at the same time maintain the data sovereignty. So, the architecture picture is evolving from what is depicted in Figure 4.2-1 to what is in Figure 7.2-1, that is with the addition of a Data Space Connector module.

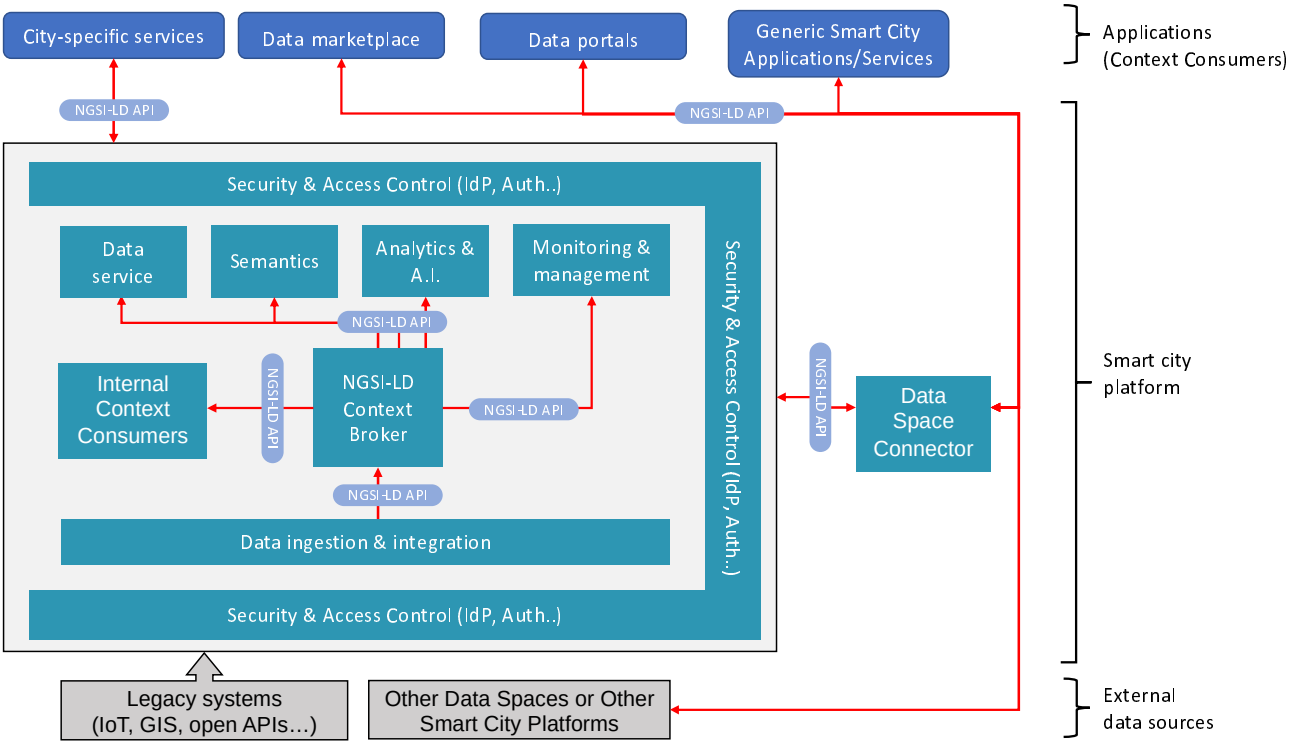

Figure 7.2-1: Smart city platform architecture as part of Data Spaces

In Figure 7.2-1 the smart city platform is evolved to be part of the Data Spaces by integrating the Data Space Connector to it without disturbing the existing functionality.

For example, a smart city platform exchanges data between energy or water utilities, citizen mobility solutions and municipal services. Each of these services is a different platform which may or may not use NGSI-LD. The services needed by smart city platforms are also predominantly cross-sector and may not follow similar data models and API standards. In this case, common API and data models will be needed to ensure smooth integration and data exchange between their data spaces. NGSI-LD can be very effective in this scenario, depicted in Figure 7.3-1. Further recommendations are described in clause 7.4.

7.3 Data Spaces overview with NGSI-LD

Data Spaces should have the following features:

1) **Data interoperability -** Data Spaces should provide a solid framework for an efficient exchange of data among participants, supporting full decoupling of data providers and consumers. This requires the adoption of a "common lingua" that every participant uses, materialized in the adoption of common APIs for the data exchange, and the definition of common data models. Common mechanisms for traceability of data exchange transactions and data provenance are also required.

2) **Data sovereignty and trust -** Data Spaces should bring technical means for guaranteeing that participants in a Data Space can trust each other and exercise sovereignty over the data they share. This requires the adoption of common standards for managing the identity of participants, the verification of the accuracy of their data and the enforcement of policies agreed upon data access and usage control.

3) **Data value creation -** Data Spaces should provide support for the creation of multi-sided markets where participants can generate value by the sharing of data (i.e. creating data value chains). This requires the adoption of common mechanisms enabling the definition of terms and conditions (including pricing) linked to data offerings, the publication and discovery of such offerings and the management of all the necessary steps supporting the lifecycle of contracts that are established when a given participant acquires the rights to access and use data.

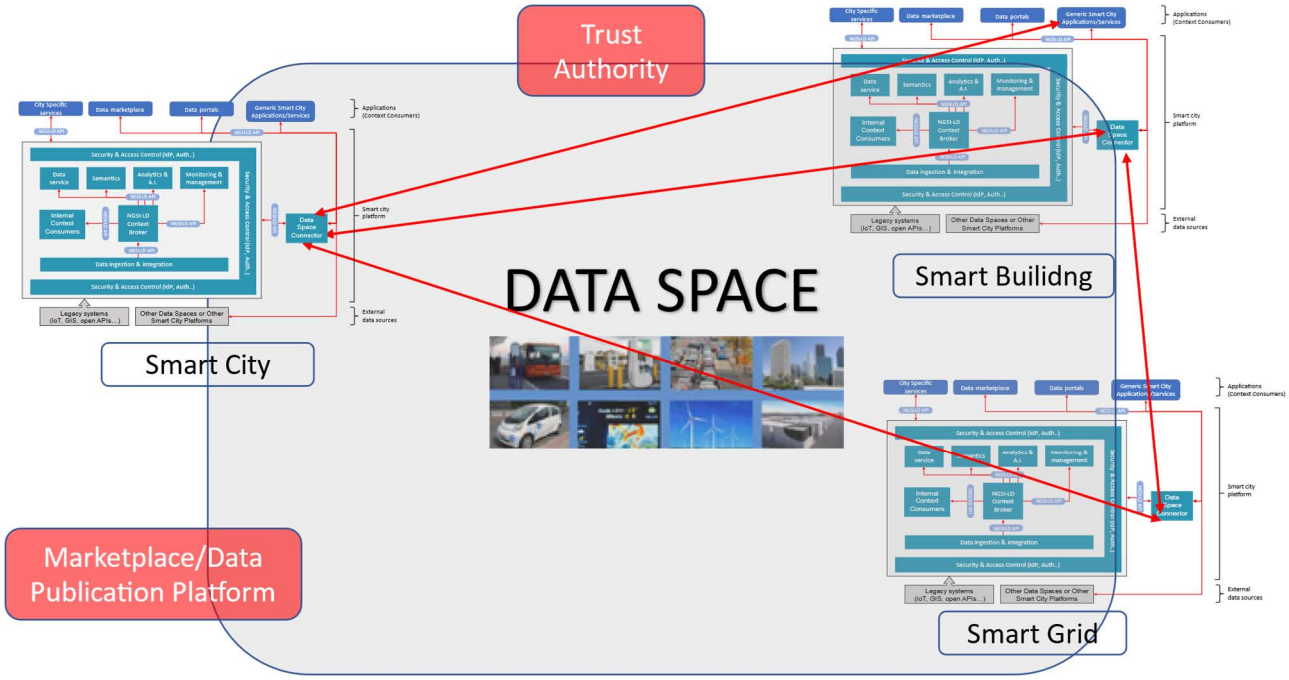

Figure 7.3-1: Data exchange in Data Spaces powered by NGSI-LD

7.4 Integration of Data Spaces with NGSI-LD

7.4.0 Introduction

NGSI-LD, which is domain agnostic, enables data exchange but needs to ensure the common data models. There are two different types of Data Spaces, as shown in Figure 7.4-1, which is copied from [i.8].

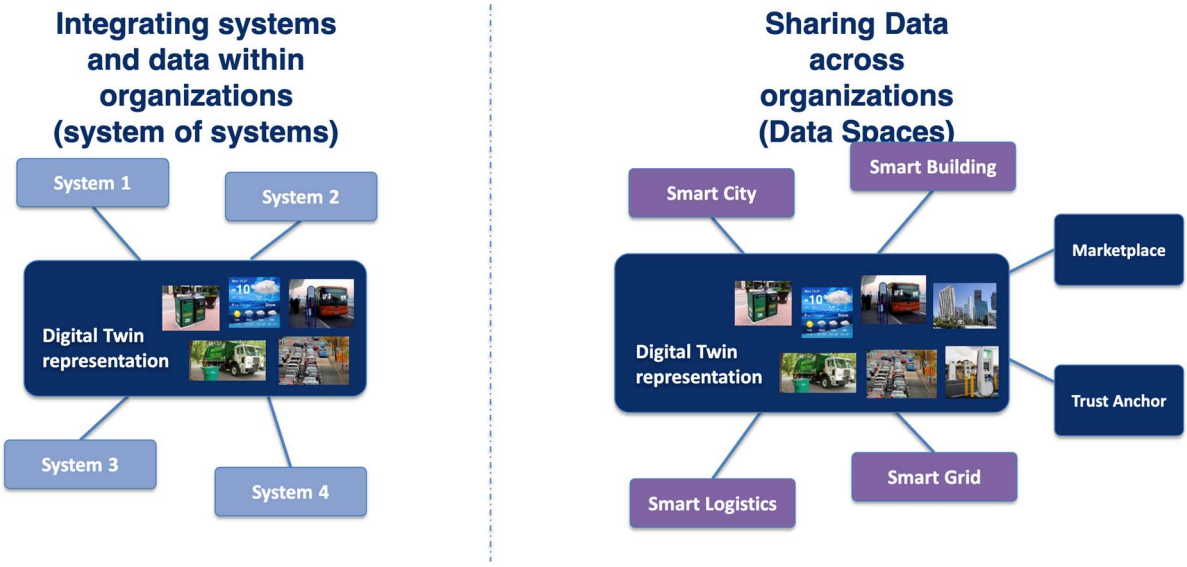

Figure 7.4-1: Essential Element of Data Spaces, copied with permission from [i.8]

In Figure 7.4-1, the data spaces are created using the System of Systems approach, where the sub-systems (most of which are already existing) in the organization are integrated to form a single data space, which will eventually become a Digital Twin representation of the organization. On the right-hand side of Figure 7.4-1, multiple organizations share such data to form a Digital Twin. For example, in a typical smart city scenario, the data is shared between Utilities, Transport and the Municipality of the city. Examples of this approach using NGSI-LD are explored in the literature [i.8].

7.4.1 Challenges

There are many challenges in creating Data Spaces, and some are listed below in increasing order of complexity:

1) There are legacy systems which need to be integrated into the Data Spaces, and some may not even have an API; and even if they have an API to share the data, they might not be in NGSI-LD format.

2) Security and privacy concerns exist regarding the systems, where the system owners want to have a sovereign right to decide the access to their data.

3) Even if the source has an API which is compatible with or using NGSI-LD API, the data model might be incompatible.

4) Even if the systems are using the NGSI-LD API and compatible data models, the relationship between *the same entities* common across the systems may be non-existent, because the systems were developed as separate data silos or the unique entities may be obfuscated for reasons of privacy.

7.4.2 Recommendations

This clause provides recommendations within the relevant scope of NGSI-LD only and does not address all the challenges mentioned in clause 7.4.1:

1) Recommendations in the white paper that contains guidelines for modelling with NGSI-LD [i.9] are of crucial importance, especially the related Recommendations 1, 2, 3 and 5 in clause 5 of the present document.

2) It is important to enable grouping of the attributes regarding the context, as explained in detail in clause 7.4.4. This recommendation is further explained in clause 7.4.3.

3) It is important to enable using interdomain (even inter-system) common contexts to create a data model where it would be possible to get the attributes in a single response from different entities in different systems. In this regard, creation of consensus ontologies such as the work motivated by the INSPIRE Directive [i.12] or the SAREF specifications [i.20] is extremely important.

7.4.3 Common attributes grouping

Grouping of attributes would be an effective way of receiving only the desired information of different entities over different systems. There are a few ways of accomplishing this grouping of attributes even though the concept is not yet defined in the NGSI-LD API:

1) Maintain the attribute groups as attributes of specially created Entities in the NGSI-LD context brokers.

2) Add attribute groups as a new concept in NGSI-LD, with the ability to create, retrieve, modify and delete attribute groups. This would be similar to the way user @context is hosted by NGSI-LD context brokers.

3) Employ the user @context to define the attribute group, and add a functionality to the NGSI-LD brokers to filter out all those attributes whose names are not a part of the user @context.

The rest of this clause is based on option 3 (using the user @context as the attribute group). An example follows, where two entities and two @context files are implemented. The URI or URL examples are not intended to be active ones. The use of "curl" as a means of sending HTTP requests is shown in the examples for convenience.

Entity 1 is defined:

```
curl -L -X POST 'http://localhost:1026/ngsi-ld/v1/entities' -H 'Content-Type:
application/ld+json' -d
{
   "id": "urn:ngsi-ld:PersonalData:PersonalData-MrX-Data-personal",
   "@context": [
         "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/parent-context.jsonld",
         "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/finances.jsonld",
   ],
   "type": "PersonalData",
   "address": {
      "value":{
      "addressCountry": "MX",
      "addressLocality": "Ciudad de Mexico",
      "streetAddress": "Colegio Franco-Ingles",
      }
   },
   "DoB": "01.01.1970",
   "qualification": "masters",
   "Height": "180",
   "kids": "3",

   "finances": {
      "gross-salary": 20000
      }
}
```

Entity 2

```
curl -L -X POST 'http://localhost:1026/ngsi-ld/v1/entities' -H 'Content-Type:
application/ld+json' -d
{
  "id": "urn:ngsi-ld:PersonalData:BankDetiails-MrX-Data-Bank",
  "@context": [
       "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/bank-finances.jsonld",
  ],
  "type": "FinancialData",

  "gross-salary": 20000,
  "bank": {
    "name of the bank": "xyz bank",
    "balance": "25000",
    "source": "http://xyz.com/account"
  }
}
```

@context file1, named "https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/finances.jsonld":

```
{
  "@context": {
    "finances": "https://smartdatamodels.org/finances"
  }
}
```

@context file2, named "https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/bank-finances.jsonld":

```
 {
    "@context": {
    "gross-salary": "https://smartdatamodels.org/gross-salary",
    "bank": "https://smartdatamodels.org/bank",
    }
  }
```

The following query can then be issued to the context broker:

```
curl –L –X GET 'http://localhost:1026/ngsi-ld/v1/entities?local=true' \
-H 'Accept: /ld+json' \
-H 'Link: <https://smartdatamodels.org/gross-salary>,
<https://smartdatamodels.org/finances>;
rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"'
```

The NGSI-LD context broker would respond with the "finances" attribute group from all the entities. Context consumers can thereby use all the attributes from different federated systems with a single query.

7.4.4 Grouping of Attribute Groups

The above approach would allow that different @contexts could be grouped and every @context could have specific key values defined under them. The context broker would only fetch the key values defined in the URL of the @context when responding to a query. In the federated approach, this could be used to get the attribute values for different entities.

In a similar way, any mechanism used for the grouping of attributes would support groups of attribute groups.

The prerequisite for this approach is that data models need to be harmonized across the data space.

8 OASC Minimal Interoperability Mechanisms

8.1 Understanding the MIM Framework

Minimal Interoperability Mechanisms (MIMs) [i.1] are defined as universal tools for achieving interoperability of data, systems, and services between cities and suppliers worldwide, championed by the Open & Agile Smart Cities (OASC) organization, a network that connects cities & communities worldwide. Related work has been initiated within the ITU-T. The mechanisms consider the different backgrounds of cities and communities and allow cities to achieve interoperability based on a minimal common ground, which is built upon an inclusive list of open standards and references.

Implementations can be different as long as the crucial interoperability points in any given technical architecture use the exactly specified interoperability mechanisms. The MIMs are vendor-neutral and technology-agnostic, meaning that anybody can use them and integrate them into existing systems and offerings.

The MIMs framework is currently divided into ten different mechanisms focusing on the key challenges for making cities smart. For each mechanism a set of specifications (called Standards and Baselines) is agreed, and a Status is assigned, marking a progressive increase in maturity of the work (Work Item, Capabilities, Specification, Governance). Agreed Specifications are available for MIM1, MIM2 and MIM3 and a brief description is given below. MIM1 is the only one that so far reached the Governance status. The remaining ones are being worked upon, being still at the status of Work Item or Capabilities:

1) MIM1 Context - MIM1 is about context information management. NGSI-LD has been selected as the mechanism of choice for MIM1, as it specifically deals with management of context information.

2) MIM2 Data Models - MIM2 is about Guidelines and catalogues of minimum common data models in different verticals, to enable interoperability for applications and systems among different cities. The specification refers to many standard data models initiatives. These are harmonized representation formats and semantics that will be used by applications both to consume and to publish data. NGSI-LD compliant data models defined by the TM Forum, OASC, IUDX and FIWARE are some of the recommended specifications.

3) MIM3 Contracts - MIM3 is about the data market places and the management layer that allows stakeholders:

   a) To provide data along with relevant information about its content and quality and any terms and conditions for use.

   b) To provide data processing services along with relevant information and terms and conditions for using the services.

   c) To find and access the data and data processing services and other services they need and to be able to gain relevant insights into what those data streams/data processing services/data applications consist of and how valuable they can be.

Though there is no direct reference to the NGSI-LD in the specifications of MIM3, the Business API Ecosystem framework [i.10], which is fully compatible with NGSI-LD, has been recommended and quoted as the framework fulfilling all the capabilities of the MIM3 specifications.

8.2 Analysis for NGSI-LD

Further analysis shows there are no specific recommendations needed for the NGSI-LD for MIM1 and MIM2, but for MIM3, the following is recommended.

- Provide an API to provide the usage metrics; specifically:

  1) Context brokers need to implement the collection of metrics for the number of times Entities, Attributes and contexts have been accessed/queried.

  2) An API needs to be specified to retrieve the usage metrics for a given entity or attribute, or context.

The metrics will help smart city administrations to understand the usage of the data and also provide metrics for various purposes. These metrics can be used to build a data economy for data providers. These metrics can also be used by Marketplaces to build revenue streams and track them.

9 GIS tools integration

9.0 Introduction

There are many definitions of smart cities, and one of them, from "How GIS Supports the Planning and Development of Smart Cities" [i.11] is as follows:

"a data-driven urban environment aimed at sustainability, transparency and efficiency, driven by an ICT enabled model rendering a visual framework, and a seamless use of disruptive technologies in various application scenarios, served by an intelligent community framework".

One of the components used often in smart cities is a GIS system which essentially provides various data points in the city in association with a Geo Spatial mapping. This allows authorities and citizens to visualize, report and address issues based on location.

There are many GIS solutions which cities have already been using even before the Smart City deployments initiated, such as ESRI's products and the German initiative called Masterportal. Mobility applications for cities need real-time visualization on map interface.

In India, often the primary interface is the so-called Situation Rooms (which are the first components to be deployed in the smart cities), which is a GIS interface showing data for the city. GIS is thus an integrated cross-sectoral platform to collect, manage, compile, analyse and visualize spatial-temporal information for sustainable urban planning, development, and management.

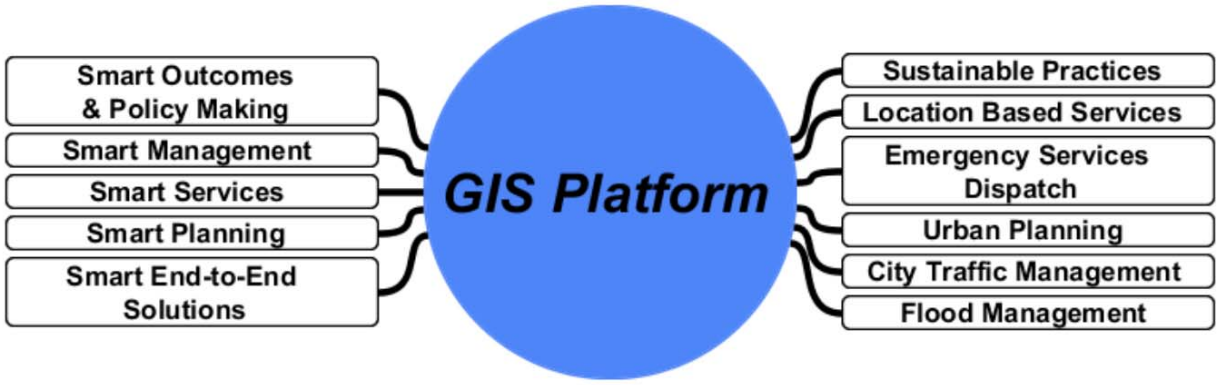

Figure 9.0-1: GIS platform, adapted from [i.11]

The trigger for similar developments in Europe has been the INSPIRE Directive [i.12], due to which various organizations started to implement their assets in GIS systems to make it available when needed. There are more than 100 cities across the world which have deployed 3rd party or self-developed GIS applications as an interface for context information management, including NGSI-LD based Context Brokers, with both real-time and non-real-time data. Some use a single-layer map base GIS and others a highly sophisticated multi-layer map GIS interface.

9.1 GIS and NGSI-LD

The ETSI NGSI-LD specification [i.2] supports GeoJSON Queries [i.17] for two-dimensional Geo Spatial data.

When considering the simple deployment shown in Figure 9.1-1, the current support from the NGSI-LD specification [i.2] is enough, but the challenges start to arise when:

1) The data needs to be exchanged between non-NGSI-LD compatible systems.

2) Topology information is needed.

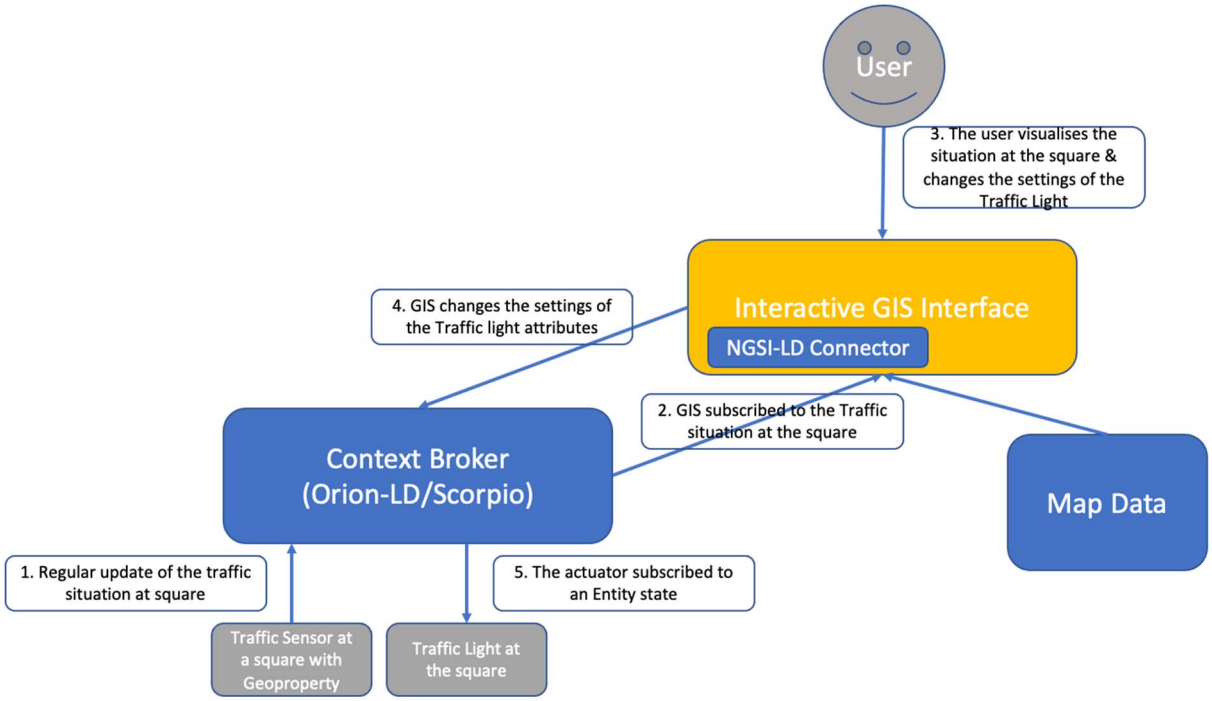

Figure 9.1-1: GIS and NGSI-LD integration

Recommendation - It would be important to consider the support for the open source TopoJSON (API [i.18]) in NGSI-LD together with GeoJSON as there will be an increase in applications where the topology information is needed for smart cities, especially mobility. TopoJSON format is supported by the software ArcGIS and QGIS [i.14].

Topology expresses the spatial relationships between connecting or adjacent vector features (points, polylines and polygons) in a GIS. Topological or topology-based data are useful for detecting and correcting digitizing errors (e.g. two lines in a road vector layer that do not meet perfectly at an intersection). Topology is necessary for carrying out some types of spatial analysis, such as network analysis. Many examples are given in QGIS and ArcGIS documentation.

Furthermore, there are many use cases which will be enhanced by supporting Geospatial topology with TopoJSON.

9.2 Compatibility with OGC API

An OGC API family of standards [i.21] is being developed to make it easy for anyone to provide geospatial data to the web. These standards build upon the legacy of the OGC Web Service standards (WMS, WFS, WCS, WPS, etc.) [i.22], but define resource-centric APIs that take advantage of modern web development practices. These standards are being constructed as "building blocks" that can be used to assemble novel APIs for web access to geospatial content. The building blocks are defined not only by the requirements of the specific standards but also through interoperability prototyping and testing.

Context Brokers are used as context information management services. Context brokers have less to do with the OGC APIs or Map data standards in the context of Smart city deployments and applications. OGC APIs are used for extracting the map data and map layer data for rendering in the GIS interface.

9.3 GeoJSON as an exchange between OGC and NGSI-LD

NGSI-LD supports all the GeoJSON geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon and GeometryCollection in the information model. Context information is provided with the geometry property in GeoJSON format if present in the queries, where it can be interpreted by most of the popular GIS systems. Therefore there are no additions that need to be recommended for the NGSI-LD specification.

9.4 Integration feasibility study with popular GIS tools

The popular GIS systems are incompatible with NGSI-LD information models, and a connector needs to be developed and maintained for integrating GIS systems with a NGSI-LD context broker. For example:

1) ArcGIS Connector was developed by ESRI to connect and access the "right-time" data from the context broker.

2) Snap4City, a platform using NGSI-LD based components, has developed connectors for ArcGIS, OpenGIS, QGIS and GeoServer applications and has been actively maintaining them.

Recommendation - Since obtaining full compatibility to various GIS applications is not possible, so it is recommended to build connectors and maintain them. The NGSI-LD information model enables and supports building of such connectors.

10 Data Sovereignty, Trust and Privacy

10.1 Overview

ETSI ISG Context Information Management (CIM) has defined NGSI-LD as a means of managing and exchanging context information (in a broad sense) between a variety of systems. The NGSI-LD API is not defining the security aspects, and the NGSI-LD brokers are generally wrapped in authentication and authorization systems to ensure that NGSI-LD APIs are accessible only to authorized and authenticated users.

Smart city platforms are built to serve citizens and enhance their services for them. It is very important to consider the security and privacy aspects when designing the NGSI-LD API and data models, together with the architecture of the platforms.

10.2 Status of Data Sovereignty with NGSI-LD

A report about security and privacy for NGSI-LD systems is ETSI GR CIM 007 [i.4] and a detailed study has been done.

In addition to the above, the following aspects should also be considered:

1) It is essential to remember that the NGSI-LD broker's core functionality is to manage the context information and not the security and privacy. Most of the time, deployments need to align with the security and privacy components of the hosting organizations and adding another layer of security within NGSI-LD would be redundant. So additional and redundant security for NGSI-LD should be avoided at the Broker level to keep the core functionality intact and efficient.

2) Context level security and privacy are recommended and need to be addressed at the context source, which does not require large changes of the NGSI-LDAPI.

10.3 MyData Study

MyData is a model whose full title is "A Nordic Model for human-centered personal data management and processing" [i.19]. In this clause important recommendations stemming from a study of MyData are reported. MyData refers:

1) to a new approach, a paradigm shifts in personal data management and processing that seeks to transform the current organization-centric system into a human-centric system;

2) to personal data as a resource that the individual can access and control. Personal data that is not under the respective individual's control cannot be called MyData.

Figure 10.3-1: Personal data is used everywhere and linked to multiple contexts (from [i.19])

The aim is to provide individuals with the practical means to access, obtain, and use datasets containing their personal information, such as purchasing data, traffic data, telecommunications data, medical records, financial information and data derived from various online services, and to encourage organizations holding personal data to give individuals control over this data, extending beyond their minimum legal requirements to do so.

In a nutshell, users (in NGSI-LD the "users" in the sense of MyData are Context Providers and Context Consumers) at the centre of their data give consent for sharing and are able to access their data and use it freely. This has significantly less to do with NGSI-LD API and more to do with Authentication and Authorization components and NGSI-LD Data models, but changes to the NGSI-LD information model and context brokers may nevertheless be needed.

Additionally, note that OASC MIM4 (Trust) is considering MyData as a reference for defining the MIM4 specification. It is recommended that NGSI-LD API should take steps towards providing an information model for Context Providers to enable data access consents.

10.4 Recommendations

10.4.0 Introduction

This clause and its sub-clauses give recommendations for using the context to group the attributes (as explained in clause 7.4.4), to filter the query response to address privacy concerns. Also, Context-Based Access Control (CBAC) can be used by the proxy to allow the query from the context consumer, based on the access control rules to the entity, and role. Recommendations include architectural aspects. The URI or URLs used in the examples below are provided as placeholders and do not refer to active hyperlinks.

10.4.1 Grouping attributes

The following example shows the context provider creating an entity.

```
curl -L -X POST 'http://localhost:1026/ngsi-ld/v1/entities' -H 'Content-Type:
application/ld+json' -d  {
   "id": "urn:ngsi-ld:PersonalData:PersonalData-MrX-Data-personal",
   "@context":
      [
         "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/parent-context.jsonld",
         "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/doctor-sport-trainer.jsonld",
         "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/spouse.jsonld",
         "https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/list-of-friends.jsonld"
   ],
   "type": "PersonalData",
   "address": {
      "value":{
         "addressCountry": "MX",
         "addressLocality": "Ciudad de Mexico",
         "streetAddress": "Colegio Franco-Ingles",
      },
   },
   "DoB": "01.01.1970",
   "qualification": "masters",
   "Height": "180",
   "weight": "104",
   "kids": "3",
   "BloodPressure": {
      "value":{
         "SYSTOLIC mm Hg ": "120",
         "DIASTOLIC mm Hg ": "85"
      },
      "source": "https://omronhealthcare.com/blood-pressure/"
   },
   "DailyActivity": {
      "value":{
         "steps per day": "10000",
         "kcal per day": "2500"
      },
      "source": "http://garmin.com/watch"
   },
   "salary": "2000",
   "friend in Finland": "X1",
   "friend in Germany": "X2"
}
```

In the above example, the attributes are to be grouped under different groups. The intention is to make Context Broker create a response based on the @context mentioned in the query and filter out the rest. In the above example, four different @contexts are mentioned.

1) Context - parent-context

   https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/parent-context.jsonld

   will have:

```
   {
     "@context": {
       "address": "https://smartdatamodels.org/address",
       "DoB": "https://smartdatamodels.org/DoB",
       "qualification": "https://smartdatamodels.org/qualification",
       "Height": "https://smartdatamodels.org/dataModel.Building/Height",
       "kids": "https://smartdatamodels.org/dataModel.Building/kids"
     }
   }
```

2) Context - doctor-sport-trainer

https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/doctor-sport-trainer.jsonld

will have:

```
{
  "@context": {
    "BloodPressure": "https://smartdatamodels.org/BloodPressure",
    "DailyActivity": "https://smartdatamodels.org/DailyActivity"
  }
}
```

3) Context - spouse

https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/spouse.jsonld

will have:

```
{
  "@context": {
    "salary": "https://smartdatamodels.org/salary"
  }
}
```

4) Context - list-of-friends

https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/list-of-friends.jsonld

will have:

```
{
  "@context": {
    "friend in Germany": "https://smartdatamodels.org/friend-in-Germany",
    "friend in Finland": "https://smartdatamodels.org/friend-in-Finland"
  }
}
```

Crucially, the Context Broker would need to filter the response based on the key values in the URL of the @context. This feature is not present in the current NGSI-LD specification [i.2], and would need to be implemented. The example is illustrated below.

So, for example, one query could be as below:

```
curl -L -X GET 'http://localhost:1026/ngsi-ld/v1/entities?local=true' \
-H 'Accept: application/ld+json' \
-H 'Link: < https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/parent-context.jsonld>;
rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"'
```

And, if multiple links in header would be allowed, for giving multiple @contexts, the query would become:

```
curl -L -X GET 'http://localhost:1026/ngsi-ld/v1/entities?local=true' \
-H 'Accept: application/ld+json' \
-H 'Link: < https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/parent-context.jsonld>, <
https://raw.githubusercontent.com/smart-data-models/dataModel.AllPersonalData/master/doctor-
sport-trainer.jsonld>;
rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"'
```

And the response from Context Broker would be (using the filtering based on what keys are present in the given @contexts):

```
{
  "@context": " https://raw.githubusercontent.com/smart-data-
models/dataModel.AllPersonalData/master/parent-context.jsonld",
  "id": "urn:ngsi-ld:PersonalData:PersonalData-MrX-Data-personal",
  "type": "PersonalData",
  "address": {
    "addressCountry": "MX",
    "addressLocality": "Ciudad de Mexico",
    "streetAddress": "Colegio Franco-Ingles",
    "type": "PostalAddress"
  },
  "DoB": "01.01.1970",
  "qualification": "masters",
  "Height": "180",
  "kids": "3"
}
```

Note that Context Brokers do not support multiple links in the Link headers and this needs to be considered, as well.

10.4.2 Data Provider has full usage control in real-time

With the above recommended approach, the Context provider can change the key values under different @contexts to provide the parts of the data in the entity based on the Request or Query to different types of Context consumers. Context providers can create new groups or destroy the existing groups to change the way the data can be accessed by Context consumers.

The provider (Data Provider or Data Owner), also with the help of the "Resource:jsonldContexts/" API (clause 6.29 of the NGSI-LD API specification [i.2]), can change the key values under the @context.

This mechanism will serve to address the Data Sovereignty aspect, which is one of the cornerstones of e.g. Gaia-X initiatives. This gives the power to rearrange the attributes to be filtered out based on the Context provider's preference. Providers do not have to wait for the policy changes to be done by the Administrator to Data Access components like PEP and PDP.

10.4.3 Data access architectural recommendations using CBAC

To effectively implement the described grouping of attributes under @context, the PEP and PDP proxies have to be configured to work with Context-Based Access Control (CBAC). Some types of the PDP (KeyCloak) support CBAC (further detailed study is needed to explore CBAC).

The proxy extracts the @context in the link header of the query from the context consumer and identifies the context consumer and the role. The proxy checks with the PDP where the context-based access control, entity level access control and Role-Based Access Control (RBAC) policies are defined. Depending on the response from the PDP the PEP proxy may forward the query to Context Broker.

Figure 10.4.3-1 is a diagram describing the flow.

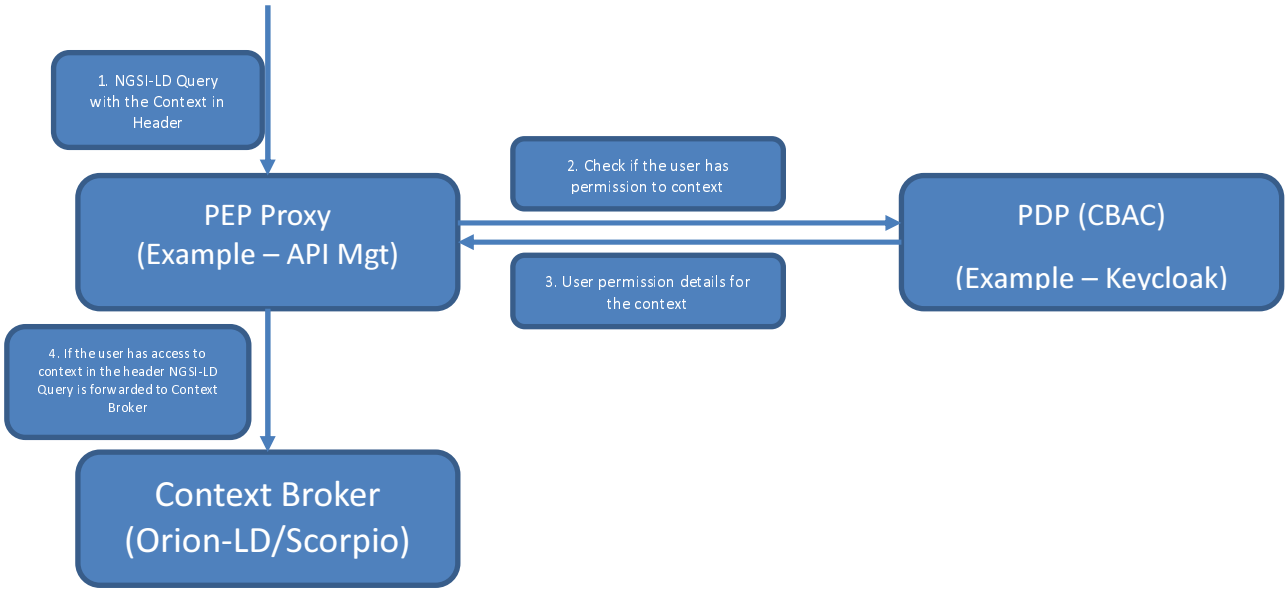

Figure 10.4.3-1: Flow of access control via proxies

10.5 Summary of recommendations

Recommendation 1	Implementation of attributes filtering in Context Brokers
Recommendation 2	Allowing multiple links in link header
Recommendation 3	Grouping attributes under @context and hosting it
Recommendation 4	Data access layer with the capabilities of CBAC

11 Linked Open Data for smart cities

11.1 Review of Semantic Web Standards

11.1.0 Introduction

Linked Open Data (LOD) is one of the services of smart cities that exploit semantic technology. In a smart city, the purpose of LOD is to build the linked data of the various smart city domains. Based on semantic web standards, all of the data in the smart cities could be connected to each other, regardless of domain, based on URIs. In this clause, LOD construction cases are described and analysed, and it is meaningful to analyse the current trends of LOD systems in smart cities and requirements for the LOD platform in the case of NGSI-LD. The goal is to compile a simple guide to getting started with LOD and NGSI-LD in smart cities. This clause highlights some potential approaches for working with LOD and then some of the considerations when attempting to bring NGSI-LD data into LOD.

Generally speaking, LOD services can more easily obtain and utilize different information thanks to the links among data. Usually, the data sets exist individually for each vertical application. For example, a data set that provides information about subway stations and a data set that provides information on air quality measurements at subway station do not have any relationship with each other. Therefore, in order to develop a service that provides integrated information on subway stations using these two sets, a process of merging heterogeneous data by the developer is required.

But, since LOD service is provided based on identifiable entities, it is easier to obtain more diverse information through a single URI. For example, the data set that provides information about subway stations and the data set that provides air quality information of subway stations are both created around subway stations. Therefore, the same URI should be given to the subway station being used in both data sets, and the two data can be linked. In this way, information of various data sets provided by an LOD portal can be more easily obtained through a single URI.

As the main technology of LOD is semantics, reviewing the existing specification and services could be the starting point to integrating with the NGSI-LD data model.

11.1.1 W3C Semantic Web Standards (SWS)

The Semantic Web [i.23], the technical specification by the World Wide Web Consortium (W3C), is machine-understandable based on an ontology including terms, complex meanings, and relationships with minimum human input. For the "Web of data" to come true, which is the goal of the Web, the Semantic Web needs access to a huge amount of linked data so that computers do more useful work and develop systems that can support trusted interactions over the network. The linked data of the Semantic Web can be created and stored on the Web, based on common formats, vocabularies, and query language, such as Web Ontology Language (OWL), Resource Description Framework (RDF), or Simple Knowledge Organization System (SKOS) and SPARQL.

11.1.2 Linked Open Data (LOD)

Linked Open Data is a way of publishing structured data that allows metadata to be connected and enriched so that different representations of the same content can be found, and links made between related resources. The link between data makes it easy to access related information via LOD. LOD is a technology exposing, sharing, and connecting the data existing on the web. As every data is linked by URI and RDF, the "linked data" means the linked data is the meaningful data connected by a link. Through the LOD service, it is possible to provide services such as linked data visualization, LOD publication, SPARQL endpoint to query data, or ontology distribution.

The main features of LOD are:

- All data constructed in LOD format can be easily accessed and utilized by anyone through a unique URI assigned to each data.
- All information is defined in RDF format and accessed through HTTP.
- To publish the LOD data in a machine-readable data format such as RDF, N3, and nTriple.
- LOD service uses SPARQL (SPARQL Protocol and RDF Query Language) to freely query and acquire desired data in the various types of data format user wants.
- LOD service makes it easier to acquire and utilize a variety of information through the connection between data.
- It provides connection information with external data as well as internal data.

11.2 Existing LODs Platforms

11.2.1 Europeana Pro

Europeana Pro (https://pro.europeana.eu/page/linked-open-data) is the LOD web portal for digitized cultural heritage collections across Europe. The metadata for all the objects in the Europeana portal is open, in that it is all licensed under the Creative Commons Universal Public Domain Dedication under the terms of the Data Exchange Agreement and can be freely downloaded via the APIs or smartphones. Once users search any collection, Europeana provides the metadata or contextual information about that with a picture so users can access digitalized museum collections of various heritage institutions. The data is represented in the Europeana Data Model (EDM) and all Europeana datasets can be explored and queried through the SPARQL API. The EDM OWL ontology is accessible through content negotiation, but it is also directly available.

11.2.2 DBPedia

DBPedia (https://www.dbpedia.org/about/) is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects. DBpedia data is served as Linked Data, which is revolutionizing the way applications interact with the Web. This structured information resembles an open knowledge graph (OKG) which is a special kind of database that stores knowledge in a machine-readable form and provides a means for information to be collected, organized, shared, searched, and utilized. DBpedia provides LodLive [i.24], which is the project for Linked Data standards such as RDF and SPARQL to browse RDF resources.

It has the following features:

- A knowledge base that extracts structured data from Wikipedia and stores it in Linked Data (RDF) format.
- Provides a great interface to check the relationship between events visually.
- Effectively used for some technology such as question-and-answer systems, text annotations, and entity name recognizers.
- Support for the DBpedia ontology and SPARQL endpoint to query with browser and various interfaces.
- Provide the triple pattern fragments, the public web service interface, the REST API.
- DBpedia Databus, the data cataloguing and versioning platform, to deploy the structured datasets.

11.2.3 Bio2RDF

Bio2RDF (https://bio2rdf.org/) is an open-source project that uses semantic web technologies to build and provide the largest network of biological interlinked life science data. Bio2RDF provides the storage for linked RDF data and shares the common ontology with URI. Bio2RDF defines a set of simple conventions to create RDF(S) compatible Linked Data from a diverse set of heterogeneously formatted sources obtained from multiple data providers.

11.2.4 Seoul Open Data Plaza LOD service

Seoul is making efforts to improve public interest, work efficiency, and transparency by opening, sharing, and communicating public data to the private sector through the Seoul Open Data Plaza LOD service (http://lod.seoul.go.kr/home/) in Korea. Seoul is disclosing various data through the Seoul Open Data Plaza (https://data.seoul.go.kr/). Especially, the Seoul Open Data Plaza LOD service selects highly useful data sets from among the data sets provided by the Seoul Open Data Plaza and provides them in the form of LODs. In addition, to ensure the real-time performance of the data provided as LOD, the data is checked for changes on a regular basis to reflect the latest data. The latest data can be provided to users and developers through real-time connection with the data set of the Seoul Open Data Plaza, and these LOD services can be provided not only through a separate LOD website but through the Seoul Open Data Plaza.

The Seoul Open Data Plaza LOD service has the following characteristics:

- Can freely use data without any conditions, such as authentication key to access the Open API, by using the URI assigned to the data.
- Can obtain and utilize linked data provided by Seoul Open Data Plaza.
- LOD service provides connection information between data provided by Seoul Open Data Plaza and external data. For example, it also provides information that the city hall station of the data set provided by the Seoul Open Data Plaza and the city hall station provided by the Korea Facilities Safety Corporation are city hall stations with the same meaning.
- The search service provides the ability to search for data built through the Seoul Open Data Plaza LOD platform. Search can be performed through the search bar at the top, which performs a search for all LOD-type data provided by platform and displays the results on the screen.
- Every URL can be accessed and can browse through the web page or download data.

There are total of 8 Seoul Open Data Plaza LOD service use cases, such as the National Library of Korea LOD, industrial property LOD, and Korean history LOD. A SPARQL endpoint supports the GET method in accordance with the W3C standard and supports queries through programs other than web pages by crafting queries using the GET method as, for example, shown below:

- http://lod.seoul.go.kr/sparql?query=select+%3Fs+%3Fp+%3Fo+where+%7B%0D%0A%3Fs+%3Fp+%3Fo+.%0D%0A%7D+limit+50+&type=html&request_method=get.

11.3 LOD with NGSI-LD

11.3.1 Implementing LOD service with NGSI-LD

In this clause, a conceptual framework is described for the goal of publishing NGSI-LD data into LOD services. So, the goal is to allow to retrieval of information from the SPARQL endpoint and the publication of all LOD data with NGSI-LD data. The NGSI-LD information data model is conceptually based on the property graph model, but mapping to triple structured data is required to support NGSI-LD data to LOD. The concept is based on a Semantic Module that constructs processes to support the activity of data that can be disclosed on the web and converts NGSI-LD to build a data web for efficient use of NGSI-LD data.

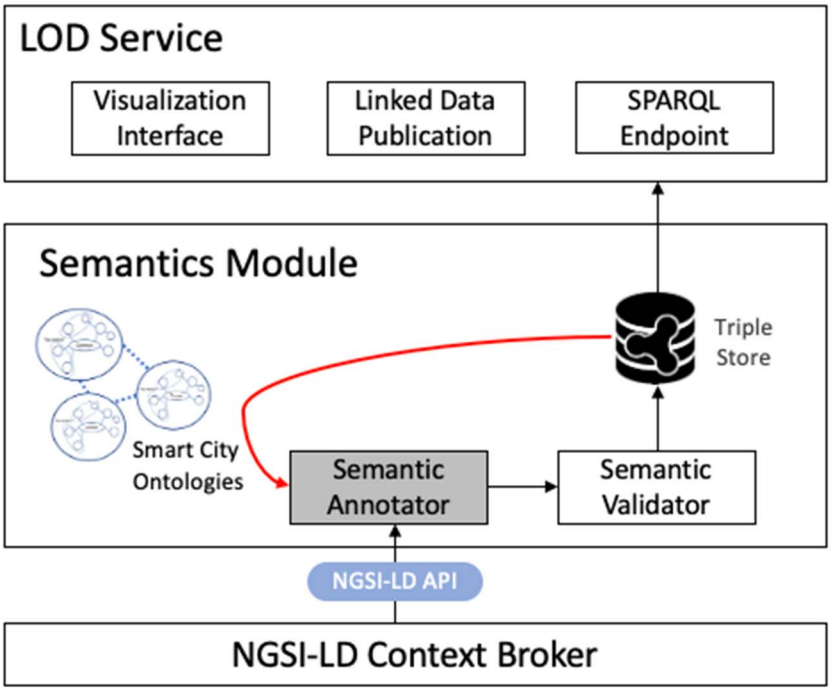

Figure 11.3.1-1: Overview of LOD with NGSI-LD

The most important part, Semantic Annotator, performs several steps that are required to convert NGSI-LD data to the LOD data before connecting with the LOD service:

- Step 1: Identify LOD construction target data information such as entity structure and type of NGSI-LD, and check whether the target data can be disclosed.

EXAMPLE 1:

```
{
   "id":"urn:ngsi-ld:ParkingLot:01",
   "type":"ParkingLot",
   "status": {
      "type": "Property",
      "value": "open"
   },
   "availableSpotNumber": {
      "type": "Property",
      "value": 40
   }
}
```

Figure 11.3.1-2: Example of LOD construction target data

- Step 2: Check if there are ontologies to be used inside the Semantics Module, that are suitable and recyclable for the data model to be constructed out of the existing ontology.

EXAMPLE 2: Analyse the vocabulary such as the name of the attribute used in the data to be constructed to determine the suitability of the ontology.

```
<owl:Class rdf:about="http://www.example.com/parking#ParkingLot">
   <rdfs:subClassOf rdf:resource="http://www.example.com/common#Zone">
</owl:Class>
<owl:ObjectProperty rdf:about="http://www.example.com/parking#hasParkingLotStatus">
   <rdfs:subPropertyOf rdf:resource="http://www.example.com/common#hasStatus">
   <rdfs:domain rdf:resource="http://www.example.com/parking#ParkingLot">
   <rdfs:range rdf:resource="http://www.example.com/parking#ParkingLotStatus">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.example.com/parking#hasParkingLotStatus">
   <rdfs:domain rdf:resource="http://www.example.com/parking#ParkingLotStatus">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string">
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about="http://www.example.com/parking#hasAvailableNumberOfSpots">
   <rdfs:subPropertyOf rdf:resource="http://www.example.com/common#hasProperty">
   <rdfs:domain rdf:resource="http://www.example.com/parking#ParkingLot">
   <rdfs:range rdf:resource="http://www.example.com/parking#AvailableParkingSpots">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.example.com/parking#hasAvailableNumberOfSpots">
   <rdfs:domain rdf:resource="http://www.example.com/parking#AvailableParkingSpots">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#unsignedInt">
</owl:DatatypeProperty>
```

Figure 11.3.1-3: Example part of ontology

- Step 3: Select or define ontologies and conceptual model that can best represent the data to be LOD and store ontologies in triple store.

EXAMPLE 3: Conceptual model: hierarchies of classes, object properties, data properties, etc.

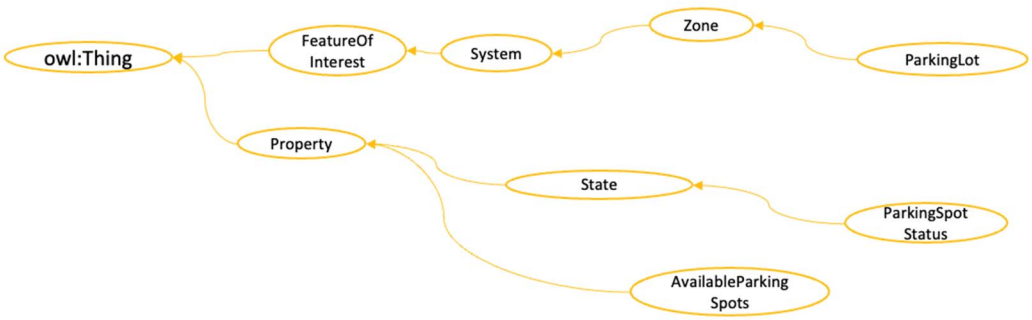

Figure 11.3.1-4: Example part of conceptual model

- Step 4: Define mapping rules for converting LOD construction target data into triple data based on the ontologies and conceptual model

EXAMPLE 4:

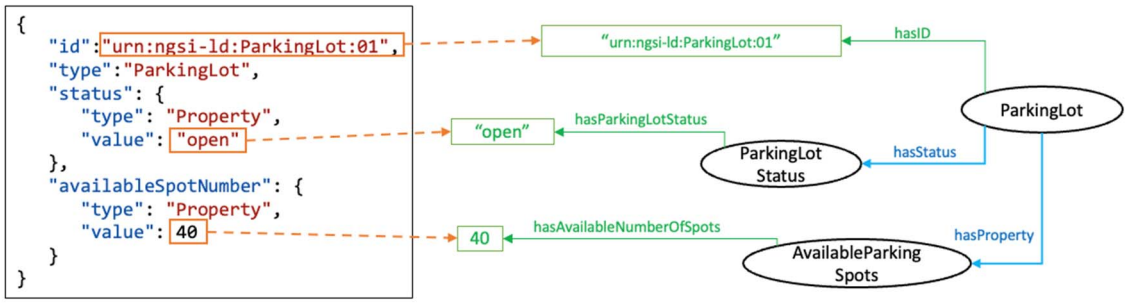

Figure 11.3.1-5: Example of mapping rule

- Step 5: Create instances by performing data transformation according to mapping rules

EXAMPLE 5: Convert NGSI-LD data to triple structured data as below.

```
<owl:NamedIndividual rdf:about="http://www.example.com/parking#ParkingLot_01">
   <rdfs:type rdf:resource="http://www.example.com/parking#ParkingLot">
   <common:hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">urn:ngsi-ld:ParkingLot:01</common:hasID>
   <hasStatus rdf:resource="http://www.example.com/parking#ParkingLotStatus_01">
   <hasProperty rdf:resource="http://www.example.com/parking#AvailableParkingSpots_01">
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.example.com/parking#ParkingLotStatus_01">
   <rdfs:type rdf:resource="http://www.example.com/parking#ParkingLotStatus">
   <hasStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string">open</hasStatus>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.example.com/parking#AvailableParkingSpots_01">
   <rdfs:type rdf:resource="http://www.example.com/parking#AvailableParkingSpots">
   <hasAvailableNumberOfSpots rdf:datatype="http://www.w3.org/2001/XMLSchema#unsignedInt">40</hasAvailableNumberOfSpots>
</owl:NamedIndividual>
```

Figure 11.3.1-6: Example of instance with OWL format

11.3.2 Recommendations

The following is recommended when implementing LOD service with NGSI-LD.

1) Start with a gradual data model that can be expanded and modified according to the characteristics of the data, and rather than creating a data model with high completeness, gradually expand the data model by reflecting the characteristics of the data set.

2) A consistent URI scheme should be applied to all data issued through the LOD, and a URI-given resource could be used to identify data and link it with other data.

3) Although this clause provides only an example of LOD data in OWL format, the publication of LOD services could provide various formats such as RDF/XML, Turtle, JSON-LD, and N3.

4) As JSON-LD is one of the possible serialization formats for LOD with NGSI-LD, but the RDF is the default way to model property graph such as NGSI-LD, and NGSI-LD information data model is also based on RDF grounding, RDF reification is recommended to support a SPARQL endpoint in the LOD service.

Annex A: Injecting NGSI-LD data to a CityGML-based 3D representation

The main idea of this annex is to check how is it possible to inject NGSI-LD data into a 3D representation based on the CityGML format. The goal here is not to map NGSI-LD attributes to CityGML objects or to map CityGML objects to NGSI-LD attributes. Granted that the smart city platform is compliant with NGSI-LD, and that the city also disposes of a 3D presentation based on the city GML specification, the goal here is to use NGSI-LD data within the 3D representation.

The CityGML standard defines a conceptual model and exchange format representing, storing, and exchanging virtual 3D city models. It facilitates the integration of urban geodata for a variety of applications for Smart Cities and Urban Digital Twins.

Two important aspects defined in the CityGML standard will help in injecting NGSI-LD data:

- The Level of Detail: according to the CityGML standard the Level Of Details on 3D representation facilitates multi-scale modelling. In CityGML, Level of Details may have values of 0, 1, 2 or 3. Objects description are more detailed with increasing LOD with respect to their geometry.
- Dynamizers: according to the CityGML standard [i.25] a Dynamizer provides the concepts that enable representation of time-varying data for city object properties as well as for integrating sensors with 3D city models. Dynamizers are objects that inject timeseries data for an individual attribute of the city.

From NGSI-LD point of view, the notification features may be exploited to get any new data from the NGSI-LD context broker. Details, such as the Level of Detail and object CityGML IDs should be added as properties on the concerned entities. An external tool then will play the role of notification handler, data adaptation and injection into the CityGML model.

As an example, see Figure A-1, suppose that the NGSI-LD based smart city data platform is counting free parking places. This parking contains 2 floors. Free places are counted for floor 1 and floor 2 separately, as well as for the whole parking. The corresponding NGSI-LD data model is based on 3 entities: each one contains the property "freePlaces". For a CityGML adaptation, two main properties will be added for each entity: "lod" to specify the level of details and "gmlId" to identify the corresponding CityGML objects of NGSI-LD entities. According to the CityGML specification, floor entities will have the level of details 2, hence the parking entity will have the level of details 2, as well.

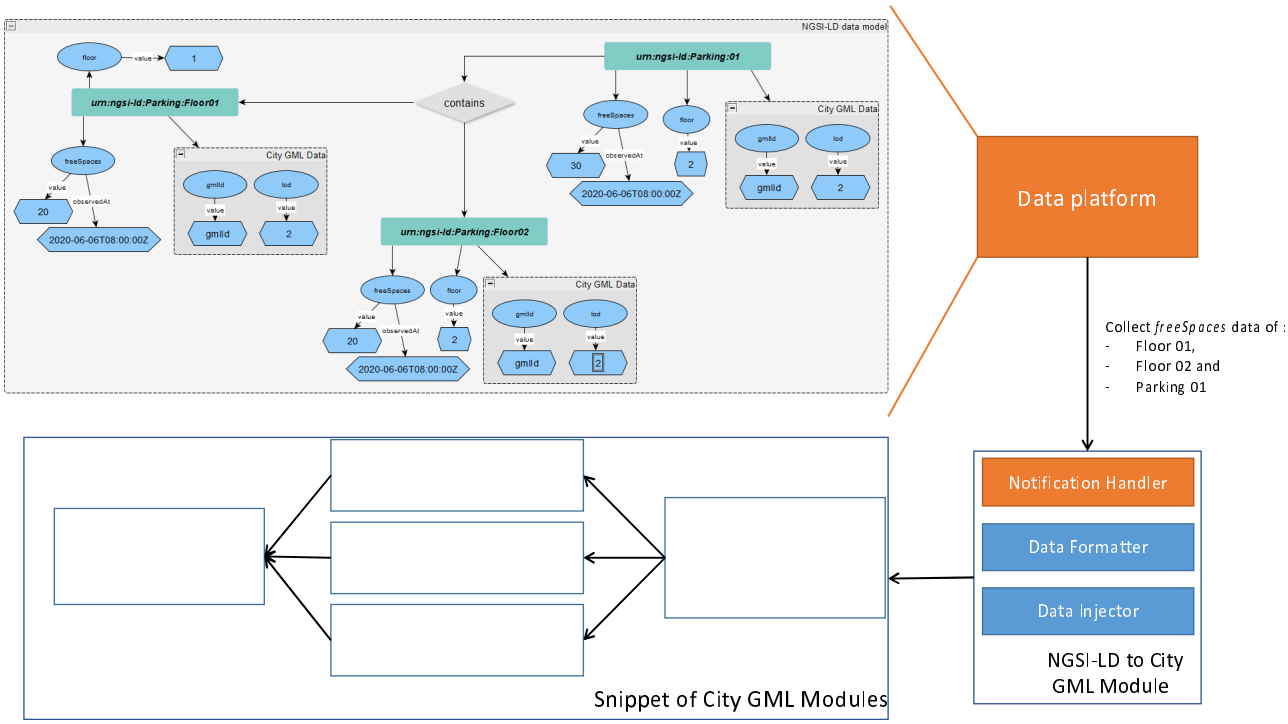

Figure A-1: Using NGSI-LD data in CityGML

History

Document history		
V1.1.1	December 2022	Publication