ETSI GR CIM 017 V1.1.1 (2022-12)

Context Information Management (CIM); Feasibility of NGSI-LD for Digital Twins

Reference

DGR/CIM-0017

Keywords

artificial intelligence, context capturing and analysis, Digital Twins, IoT

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

Foreword

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

Modal verbs terminology

Executive summary

The present document analyses the use of NGSI-LD information model and API for representation and handling of digital twins. It starts with a description of actual digital twin use cases, in several vertical domains. Then it provides an identification of relevant external technical initiatives. Finally, the present document provides concrete views on the definition of Digital Twins in an NGSI-LD ecosystem including identification of Digital Twin capabilities and life cycle stages. The final clauses analyse how, taking inspiration from actuation mechanisms, a service execution API could be defined to handle Digital Twin capabilities.

Introduction

The present document advocates the use of NGSI-LD property graphs as holistic Digital Twins, maintaining multi-level and multi-scale descriptions of complete environments, such as cities, buildings or factories. The nodes of the proposed multilevel graph stand for any real-assets, which can be a physical asset or a concept. The arcs of the graph represent relationships between these entities, which capture the physical and information structure of a system. They can capture top-down and bottom-up system composition relationships, or transversal connectors (like cables, pipes, etc.) in a distributed network-like system. A further level of description captures distributed or loosely coupled "systems of systems" as "graphs of graphs", i.e. graphs, whose "hypernodes" encapsulate other graphs). By maintaining this shared context, the graph makes it possible for different applications in these environments to locate and share their data sources on the basis of the consolidated information stored in the graph, much as the knowledge graph of as search engine does for the Web. The graph platform can play the role of a Digital Twin, in the sense of a one-stop-shop for applications operating upon these environments. The NGSI-LD specification is thus already an actor in the Digital Twins area and the present document aims at exploring its current usage, positioning within the overall Digital Twins technical landscape and providing recommendations for NGSI-LD specification evolution.

Most of the work referenced in the present document was created with the support of the following European Union Horizon 2020 research projects: No. 818036 (iFishIENCi), No. 821036 (FIWARE4WATER), No. 871319 (ADEPTNESS), No. 731993 (AutoPilot), No. 832876 (aqua-3S), No. 814918 (Fed4IoT) and French project "Cloud Continuum Souverain et Jumeaux Numériques" funded by BPIFrance under contract DOS0179607/00 - DOS0179608/00, including many contributions from members of the FIWARE® Community.

1 Scope

The purpose of the present document is to show to what extent various Digital Twin types can be realized or facilitated by NGSI-LD and to identify new features for NGSI-LD which would make it more useful for such areas of usage.

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] Grieves, Michael, et John Vickers: "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". In Transdisciplinary Perspectives on Complex Systems, edited by Franz-Josef Kahlen, Shannon Flumerfelt, and Anabela Alves, 85-113. Cham: Springer International Publishing, 2017.

NOTE: Available at https://doi.org/10.1007/978-3-319-38756-7_4.

[i.2] Gilles Privat: "Phenotropic and stigmergic webs: The new reach of networks". Universal Access in the Information Society 08/2012; 11(3):1-13., DOI:10.1007/s10209-011-0240-1.

[i.3] Gilles Privat: "The "Systems of Systems". Viewpoint in telecommunications", Orange Research Blog, September 2018.

[i.4] OWA-EPANET Toolkit.

NOTE: Available at http://wateranalytics.org/EPANET/.

[i.5] Tuegel, Eric, Ingraffea, Anthony, Eason, Thomas, et Spottswood, Stephen. (2011): "Reengineering Aircraft Structural Life Prediction Using a Digital Twin". In International Journal of Aerospace Engineering, 2011.

NOTE: Available at https://doi.org/10.1155/2011/154798.

[i.6] Boss, Birgit & Malakuti, Somayeh & Lin, Shi-Wan & Usländer, Thomas & Clauer, Erich & Hoffmeister, Michael & Stojanovic, Ljiljana & Flubacher, Björn. (2020): "Digital Twin and Asset Administration Shell Concepts and Application in the Industrial Internet and Industrie 4.0". An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper.

NOTE: Available at https://www.iiconsortium.org/pdf/Digital-Twin-and-Asset-Administration-Shell-Concepts-and-Application-Joint-Whitepaper.pdf.

[i.7] ETSI GS CIM 009: "Context Information Management (CIM); NGSI-LD API".

[i.8] ETSI GS CIM 006: "Context Information Management (CIM); Information Model (MOD0)".

[i.9] "PackML language for the control of packaging machines".

NOTE: Available at https://en.wikipedia.org/wiki/PackML.

[i.10] IEC TC 65: "Industrial-process measurement, control and automation".

NOTE: Available at https://www.iec.ch/dyn/www/f?p=103:7:14089666729290::::FSP_ORG_ID:1250.

[i.11] ISO 23247-1:2021: "Automation systems and integration -- Digital twin framework for manufacturing -- Part 1: Overview and general principles".

NOTE: Available at https://www.iso.org/standard/75066.html.

[i.12] ISO 23247-2:2021: "Automation systems and integration -- Digital twin framework for manufacturing -- Part 2: Reference architecture".

NOTE: Available at https://www.iso.org/standard/78743.html.

[i.13] ISO 23247-3:2021: "Automation systems and integration -- Digital twin framework for manufacturing -- Part 3: Digital representation of manufacturing elements".

NOTE: Available at https://www.iso.org/standard/78744.html.

[i.14] ISO 23247-4:2021: "Automation systems and integration -- Digital twin framework for manufacturing -- Part 4: Information exchange".

NOTE: Available at https://www.iso.org/standard/78745.html.

[i.15] ETSI TS 103 828: "SmartM2M; SAREF: Ontology Support for Urban Digital Twins and usage guidelines".

NOTE: Available at https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=63076.

[i.16] ISO/IEC AWI 30173: "Digital twin -- Concepts and terminology".

NOTE: Available at https://www.iso.org/standard/81442.html.

[i.17] IEC Technology report: "City information modelling and urban digital twins".

NOTE: Available at https://www.iec.ch/basecamp/city-information-modelling-and-urban-digital-twins.

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:

AASX	Asset Administration Shell Explorer
AEC	Architecture, Engineering, Construction
AI	Artificial Intelligence
AMR	Autonomous Mobile Robots
API	Application Programming Interface
AR	Augmented Reality

ARF	Augmented Reality Framework
BIM	Building Information Management
DMA	District Metered Area
DT	Digital Twin
DTDL	Digital Twin Description Language
EC	European Commission
EPANET	Environmental Protection Agency Network Evaluation Tool
F6G	Fixed 6G
GIS	Geographic Information System
HiL	Hardware in the Loop
HTTP	HyperText Transfer Protocol
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
IEC	International Electrotechnical Commission
IIoT	Industrial Internet of Things
IoT	Internet of Things
IOWN	Innovative Optical and Wireless Network
IT	Information Technology
JSON	JavaScript Object Notation
JTC	Joint Technical Committee
LD	Linked Data
LWM2M	Lightweight Machine To Machine
M2M	Machine To Machine
ML	Machine Learning
MLS	Machine Learning Service
MQTT	Message Queuing Telemetry Transport
NGSI	Next Generation Service Interfaces
OASCs	Open and Agile Smart Cities
OGC	Open Geospatial Consortium
PackML	Packaging Machine Language
QoS	Quality of Service
RAMI	Reference Architectural Model Industry
RFID	Radio Frequency IDentification
ROS	Robot Operating System
RWA	Real World Asset
SAREF	Smart Applications REFerence
SCADA	Supervisory Control and Data Acquisition
SiL	Software in the Loop
SMS	Short Message Service
SoSTs	Systems of systems Twins
ST	System Twin
STF	Specialist Task Force
STs	System-Twins
UDT	Urban Digital Twins
UI/UX	User interface/User experience
UL	Ultra Light
URI	Universal Resource Identifier
WDS	Water Distribution System
WoT	Web of Things

4 Illustrative use cases

4.0 Introduction

Clause 4 describes some of the on-going initiatives making use of NGSI-LD information model and API to define and interact with Digital Twins. The concept of Digital Twins is not elaborated here, because many variations are explained in the literature [i.1], [i.2], [i.5],[i.6], [i.11], [i.15], [i.16], [i.17].

4.1 Water distribution network

4.1.1 Usage

Management of Water Distribution Systems (WDSs) is an area where DTs have significant potential, and they are already starting to be implemented to address issues such as optimizing operation of the system, asset management and water leak localisation. However, development of DTs for WDSs can be complex, with the need for any hydraulic and/or quality model to be constantly paired with data and information from multiple sources in the physical world, including, for example, Supervisory Control and Data Acquisition (SCADA) and consumption metering systems. Transmission, conversion, storage and protection of this data are all important issues for consideration and are currently complicated by a lack of standardization. However, standardization, along with interoperability and integration are fundamental features of a successful DT. The described research integrates the WDS simulation toolkit, OWA-EPANET [i.4], within a FIWARE NGSI-LD environment.

The functionalities foreseen from that integration focus on the system in operation and include:

- **Simpler interface to a composed system:** such as exposing the "equivalent water pump" of N water pumps connected in parallel. This equivalent pump can thus be queried and configured as if it would be a real water pump.
- **Predictive analytics:** to provide a prediction of the system state in the future (T1), based on the current state (T0).
- **Outliers/anomaly detection:** to raise alerts in case the network deviates from its anticipated state.

EXAMPLE 1: By comparing the actual state of the system at T1 with the simulation T1 executed at T0.

EXAMPLE 2: By comparing the state of a subnetwork with expected state calculated using simulation based on other part of the system.

- **What-if scenario:** how will the system evolve if an action is taken starting from the current state, modifying some parameters in the model (i.e. close a valve).
- **Network optimisation:** starting from the current state, identify the optimum functioning point (i.e. reduce water pump energy consumption while maintaining requested pressure). When combined with actuation (not mandatory), this functionality provides automated management.

4.1.2 Challenges

Several challenges have been identified:

- Sharing a common data model representation between the target EPANET simulator and the NGSI-LD representation.
- Having a model which allows the selection of only sub-parts of the water network (i.e. DMA - District Metered Area level) to be used within the simulation.
- To provide a service feeding the simulator with appropriate values and configuration parameters and storing the resulting state without storing the whole simulated result, so as to avoid overloading the system storage.

4.1.3 Solution proposal

A NGSI-LD data model for water network, aligned with the definition of the EPANET hydraulic simulator has been proposed (https://github.com/smart-data-models/dataModel.WaterDistributionManagementEPANET). An overview is provided in the picture below. All parts of the network are considered of equal importance from a modelling point of view so they have all been modelled as entities, including pipes. In addition, most of the network entities are directed, so connections are made through *startsAt* and *endsAt* relationships.

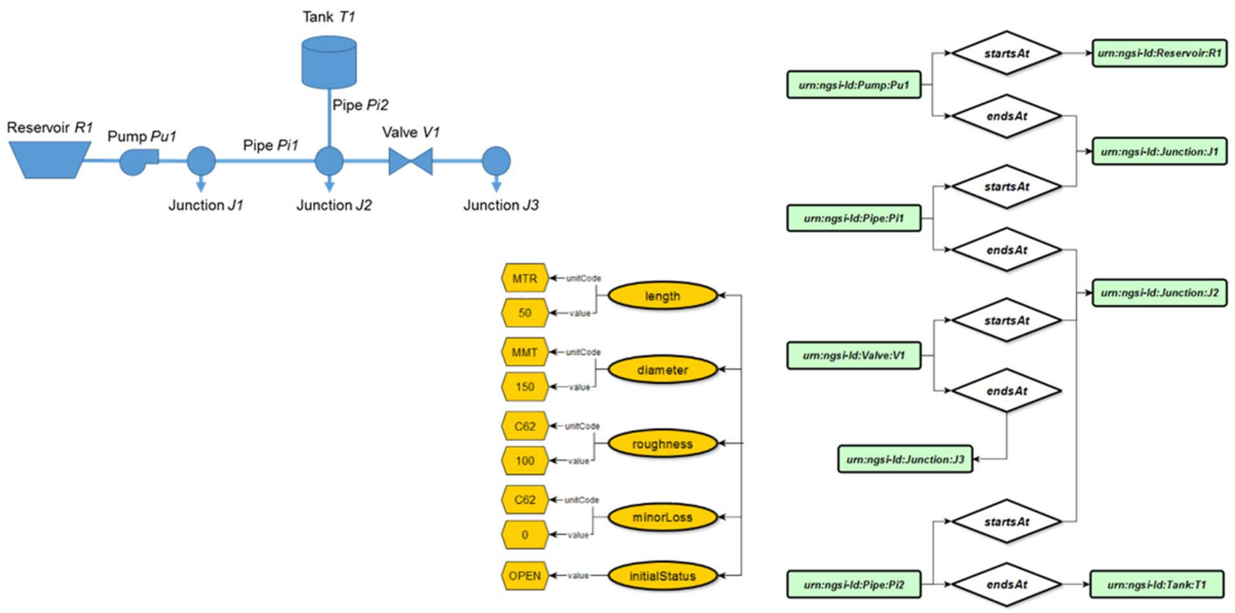


Figure 4.1.3-1: NGSI-LD model of a water distribution network

In addition, as seen on the next picture, a system composition approach has been made by defining entities representing the different DMAs. This allows a simulation to handle only one specific DMA. Any simulation is associated to a NGSI-LD entity which holds through its attributes: the simulator configuration options, possibly, a list of value to be changed change from actual values (to run a "what-if" scenario) and the list of points in the water network to be controlled so not all the network state needs to be saved.

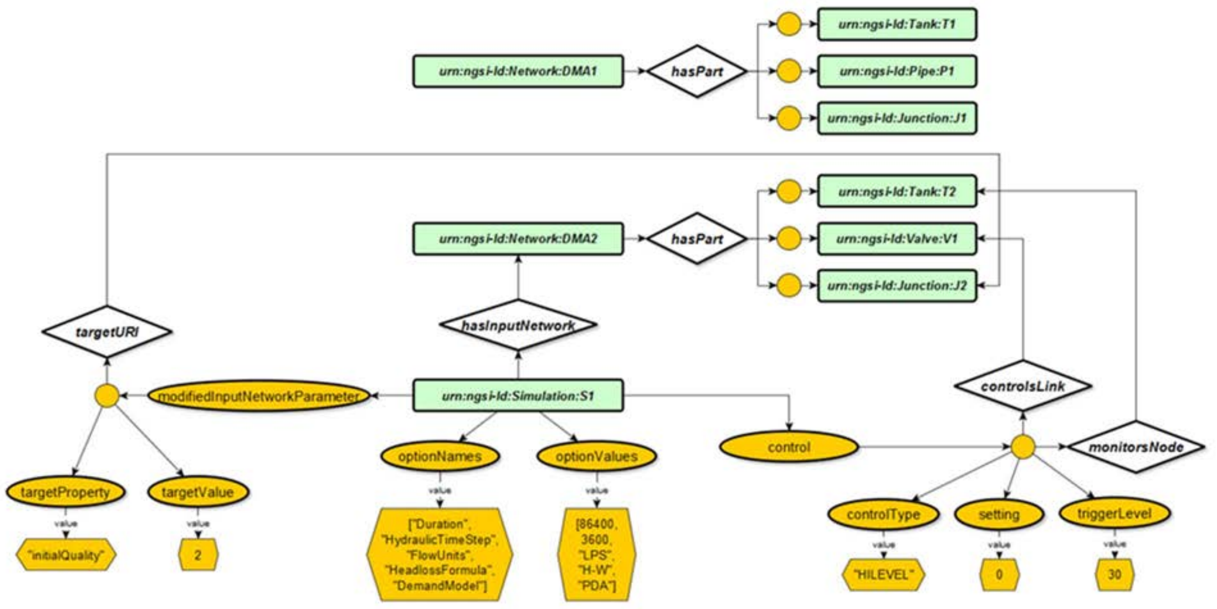


Figure 4.1.3-2: Handling of simulation parameters within a NGSI-LD representation

To facilitate the integration of the NGSI-LD context information and EPANET, a new interface has been developed. The key functionalities provided by this are:

a) translation of existing EPANET model data into the requisite NGSI-LD format;

b) posting of this information to a NGSI-LD context broker;

c) retrieval of all data necessary to generate an up-to-date network model for simulation (capturing real-time network data supplied to the context broker from other sources such as IoT platforms); and

d) as and when required, running hydraulic and quality simulations of the network.

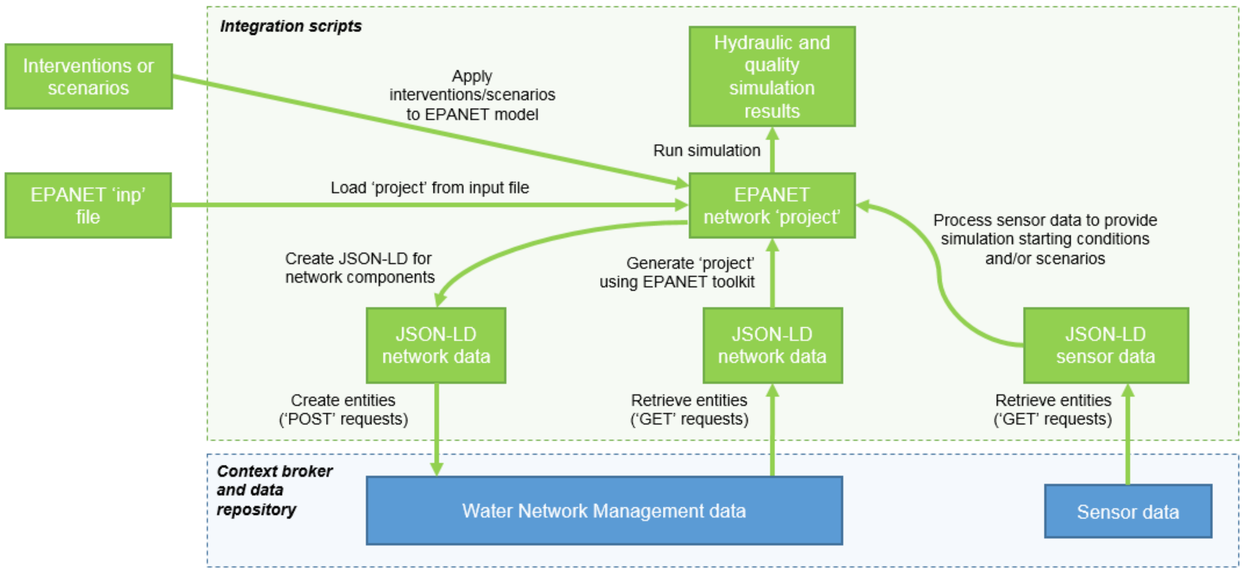

Integration scripts
Interventions or scenarios
Apply interventions/scenarios to EPANET model
Hydraulic and quality simulation results
Run simulation
EPANET 'inp' file
Load 'project' from input file
EPANET network 'project'
Process sensor data to provide simulation starting conditions and/or scenarios
Create JSON-LD for network components
Generate 'project' using EPANET toolkit
JSON-LD network data
JSON-LD network data
JSON-LD sensor data
Create entities ('POST' requests)
Retrieve entities ('GET' requests)
Retrieve entities ('GET' requests)
Context broker and data repository
Water Network Management data
Sensor data

Figure 4.1.3-3: Architecture and functionalities provided in the EPANET-FIWARE integration module

The new NGSI-LD-integrated implementation of EPANET outlined in the previous sections has been applied to a case study water distribution network from the South West of England to test and demonstrate some of the functionalities offered. The network is a small, gravity-fed system, with the EPANET model containing only one source (modelled as a reservoir), 1 005 nodes and 1 035 links. It also contains 91 household level smart meters, which provide daily water consumption measurements for individual houses. Data from these meters is posted to a Stellio context broker in accordance with the 'WaterSmartMeter' data model.

In this case study, historical smart meter data is used to adjust and extend the demand patterns in the EPANET model to capture daily variations in demand and enable more realistic simulation of hydraulic performance under historical conditions. To illustrate the impact of using smart meter data instead of the default demands defined in the EPANET input file on the hydraulic simulation results. Figure 4.1.3-4a) compares the results of pressure time series for one junction with an associated smart meter, Figure 4.1.3-4b) shows the flow rate in the pipe supplying this node.

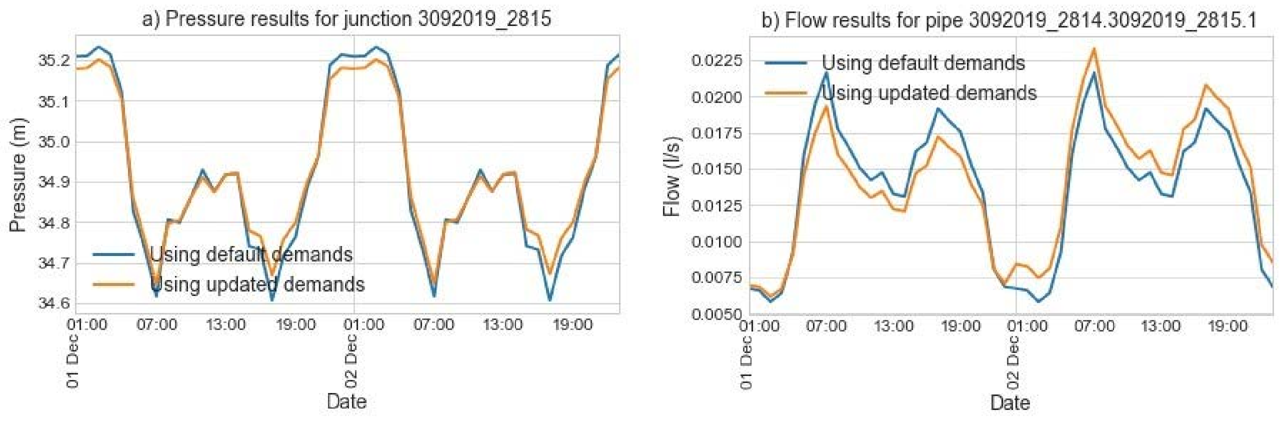


Figure 4.1.3-4: Example of hydraulic results obtained with the default network model demands and with demands updated based on data retrieved from smart meters

4.1.4 Gaps

It appeared that in its current version the NGSI-LD specification is fully relevant for definition and management of Digital Twins, even, in the context of a water distribution network made of several sub-systems. Still, some improvement could be made:

- Improved handling of system composition to avoid multiplying relationships toward all entities of a sub-system.

- Improve management of relationships to make them first class citizen as are the entities in the NGSI-LD API to allow more traditional modelling of a water distribution network. Nevertheless, handling the fact that a pipe is not directed (water can flow in both way) within a directed graph would require additional investigations.
- Provide capabilities to ease handling of a simulator within a NGSI-LD based deployment. This includes configuration of the simulator, handling of some special scenarios (e.g. What-if) and storage of the key results.

4.2 Digital Twins in robotics

4.2.1 Usage

Two basic but essential core robotics scenarios should be explored as application enablers:

- 2D Robot Navigation (e.g. autonomous robot navigation for intra-logistics).
- Pick and Place Operations (e.g. palletization, packaging, product sorting, etc.).

An advanced scenario is the 3D navigation of drones. In turn, many features of this scenario may be simplified and represented as a 2D Navigation.

EXAMPLE: Autonomous Mobile Robot for Warehouse Automation:

- **Robot Entity:** AMR (Transportation Robot) → Autonomous Navigation System (Interfaces to Automated Path Planning Module, trajectory planning and obstacle avoidance are opaque).
- **Robot Task:** Move 'Item X' from 'Loc A' to 'Loc B'.
- **EnvModel:** Layout of the factory with annotatedLocations, **Personnel:** Warehouse Operator.
- **IIoT Device:** RFID Reader; **User Interface:** Stock Transfer Order Request.

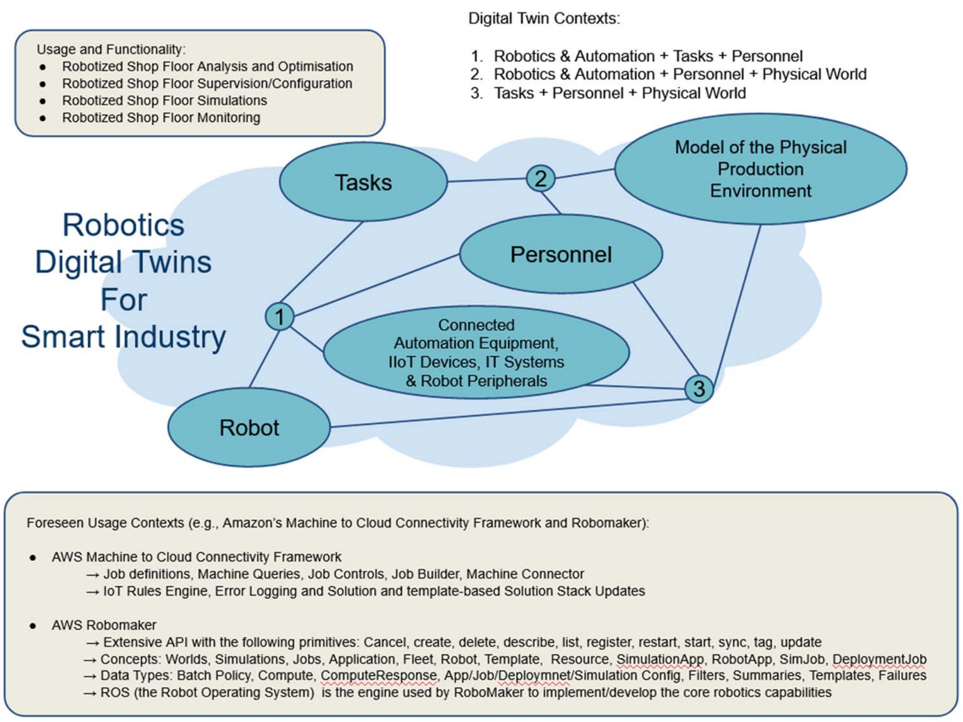

Usage and Functionality:
• Robotized Shop Floor Analysis and Optimisation
• Robotized Shop Floor Supervision/Configuration
• Robotized Shop Floor Simulations
• Robotized Shop Floor Monitoring
Digital Twin Contexts:
1. Robotics & Automation + Tasks + Personnel
2. Robotics & Automation + Personnel + Physical World
3. Tasks + Personnel + Physical World
Robotics Digital Twins For Smart Industry
Tasks
2
Model of the Physical Production Environment
Personnel
1
Connected Automation Equipment, IIoT Devices, IT Systems & Robot Peripherals
3
Robot
Foreseen Usage Contexts (e.g., Amazon's Machine to Cloud Connectivity Framework and Robomaker):
• AWS Machine to Cloud Connectivity Framework
→ Job definitions, Machine Queries, Job Controls, Job Builder, Machine Connector
→ IoT Rules Engine, Error Logging and Solution and template-based Solution Stack Updates
• AWS Robomaker
→ Extensive API with the following primitives: Cancel, create, delete, describe, list, register, restart, start, sync, tag, update
→ Concepts: Worlds, Simulations, Jobs, Application, Fleet, Robot, Template, Resource, SimulationApp, RobotApp, SimJob, DeploymentJob
→ Data Types: Batch Policy, Compute, ComputeResponse, App/Job/Deploymnet/Simulation Config, Filters, Summaries, Templates, Failures
→ ROS (the Robot Operating System) is the engine used by RoboMaker to implement/develop the core robotics capabilities

Figure 4.2.1-1: Foreseen usage and expected functionalities from a DT deployment

4.2.2 Challenges

The main challenges Identified for NGSI-based Digital Twins of Robotics Systems are:

- The software architecture of robotics systems is often a complex, monolithic architecture and largely conditioned by mechatronic requirements. Robustness and efficiency are often at a premium. Reusability, modularity, interoperability as they are understood in the IT world have been, in general, secondary aspects within the robotics domain.
- Even at higher levels of discrete control and plan execution the interfaces are extremely heterogeneous. Almost every robot manufacturer has its own framework/programming suite to develop applications and relies on an ad/hoc communication protocol to integrate them.
- The Robot Operating System has been trying (and still tries) to offer a common framework for open-source robotics application development. The adoption in real-world scenarios out of prototypes, laboratories, and experimental settings is still limited. This invites exploring the robotics frameworks developed by robot manufacturers and those developed by widespread system integrators and application simulators (Visual Components). Probably, our space to create value for robotics applications is the implementation of capabilities that easily integrate on top of these frameworks.

4.2.3 Solution proposal

The proposed solution is made of the following:

1) Smart Gateway on Top of the Native Robot Middleware.

2) NGSI-LD Compliant Digital Twin.

3) Smart Cloud for Accessible and Scalable Robotics Applications.

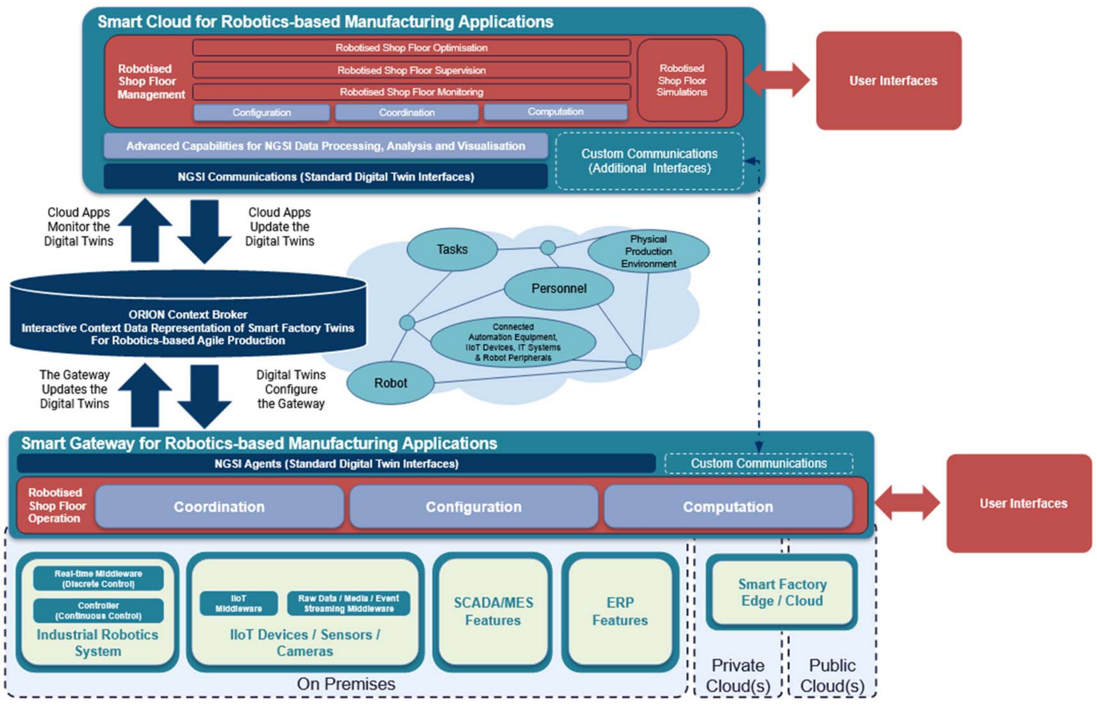

Smart Cloud for Robotics-based Manufacturing Applications
Robotised Shop Floor Management
Robotised Shop Floor Optimisation
Robotised Shop Floor Supervision
Robotised Shop Floor Monitoring
Configuration
Coordination
Computation
Robotised Shop Floor Simulations
User Interfaces
Advanced Capabilities for NGSI Data Processing, Analysis and Visualisation
NGSI Communications (Standard Digital Twin Interfaces)
Custom Communications (Additional Interfaces)
Cloud Apps Monitor the Digital Twins
Cloud Apps Update the Digital Twins
Tasks
Physical Production Environment
Personnel
Connected Automation Equipment, IIoT Devices, IT Systems & Robot Peripherals
Robot
ORION Context Broker
Interactive Context Data Representation of Smart Factory Twins
For Robotics-based Agile Production
The Gateway Updates the Digital Twins
Digital Twins Configure the Gateway
Smart Gateway for Robotics-based Manufacturing Applications
NGSI Agents (Standard Digital Twin Interfaces)
Custom Communications
Robotised Shop Floor Operation
Coordination
Configuration
Computation
User Interfaces
Real-time Middleware (Discrete Control)
Controller (Continuous Control)
Industrial Robotics System
IIoT Middleware
Raw Data / Media / Event Streaming Middleware
IIoT Devices / Sensors / Cameras
SCADA/MES Features
ERP Features
Smart Factory Edge / Cloud
On Premises
Private Cloud(s)
Public Cloud(s)

Figure 4.2.3-1: High level architecture for Digital Twins in robotics

Data: ROS Messages already define a number of properties and structures for core robotics applications that have remained constant for long. One need to adopt additional data models from industry standards, in particular those modelling real-world environments and devices and with the monitoring/configuration of automated jobs and simulations.

Security: Since the robot interacts with the physical world, every scenario that considers an application scenario that goes beyond the typical hard-coded behaviours for standalone robotics faces a number of security and safety issues.

Services beyond v1.3 of API: Robot entities should advertise their special capabilities and their current hardware state/configuration. Ideally the Digital Twin should automatically update its state when some end-effectors, sensors or peripherals are connected, changed, replaced or removed from the base robot.

4.2.4 Gaps

NGSI-LD has potential to implement powerful robotics twins at the "integrated planning and execution levels" in which context semantics play relevant roles. Objective is then to provide a common context data layer for heterogeneous robot twins based on standardized context information management.

The main gap to be filled is the integration of NGSI-LD based robotics capabilities within the native executive features of base robotics platforms. The design of criteria to conveniently monitor/sample/configure the discrete control loop of the actual robot is not straightforward and still lack mechanisms to implement smart integration behaviours. Even in simple robotics scenarios, the real-time monitoring of robot resources and capabilities as well as the maintenance of right-time synchronized representation of the robot world is a challenging task.

4.3 Use of Digital Twins in the aeronautic sector

4.3.1 Usage

The Aeronautical sector is one of the many sectors in which large amounts of data are generated and, consequently, they can be exploited by the implementation of Digital Twins, improving the performance of processes through the communication between the physical and the virtual world. This use case presents an Airport Digital Twin Reference used to improve turnaround process.

Airports are one of the infrastructures that require more organization and security protocols since they have to deal with a high density of passengers, staff, aircrafts, baggage, data, etc. In particular, flight delays are a common problem affecting airports, airlines and passengers. The lack of digitalization has made the turnaround process a bottleneck in airport operations and a common cause of delays. This process is made up of several tasks, and in most airports, their operations are coordinated through radio communications and paper forms completed by workers.

This use case shows how to improve the turnaround process thanks to context information management provided by different sources, and how to take advantage of web services and 3D representations instead of printed version forms.

4.3.2 Challenges

DTs have been very prolific in the Aeronautics sector, in fact, one of the first DT use cases was developed in this domain in the year 2011 [i.5]. Recently the interest of building DTs has increased in this sector, for both specific purposes (e.g. modelling aircraft turbines and engines), and for general ones (e.g. modelling the whole airport). The present document describes a DT of a commercial, with the objective of deploying a 2D/3D view application fed with information in (pseudo)real-time about the airport, including, the stand occupancy, flight information, turnaround events, etc., together with an operator's application that registers the flight tasks.

Examples of usage:

- Operator: 2D/3D navigation viewing the pending and completed tasks regarding the aircraft "W", placed in the stand "X", related to the flight "Y" that is scheduled to depart in "Z" minutes.
- Passenger: 2D/3D navigation following its own luggage position in the airport.

The main challenge is on data ingestion and modelling from external sources: each source provides its data in its own format, using its own protocols and standards. In the Aeronautics sector there are several widely disseminated standards (e.g. ICAO, IATA). As a consequence, in some cases it is difficult to establish relationships between entities. As an example, an API that identifies a flight with the ICAO format while another identifies it using the IATA designator.

4.3.3 Solution proposal

Figure 4.3.3-1 shows all the agents present in the Airport Digital Twin use case, including the data source/sink entities, which interact with a NGSI-LD compliant platform.

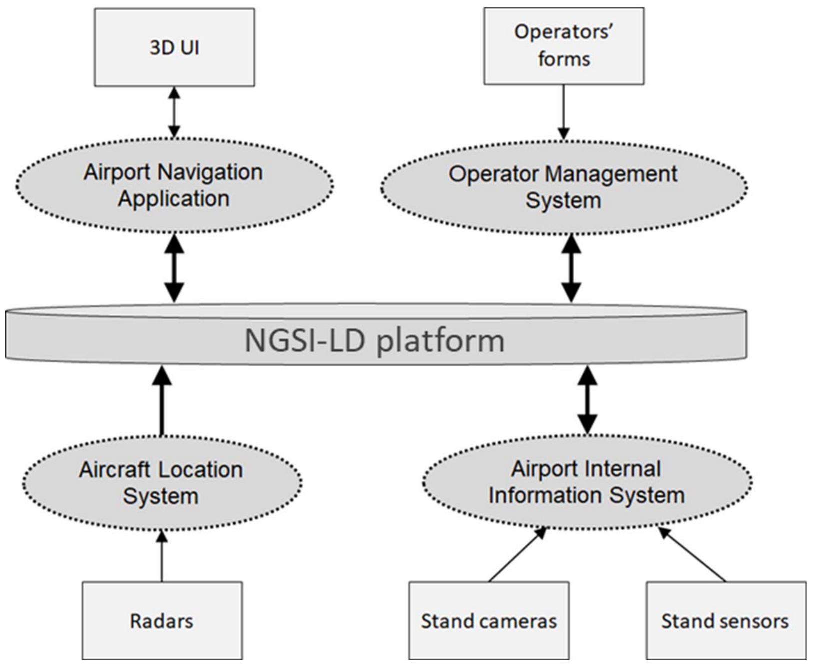


Figure 4.3.3-1: Context Information Exchange between Agents and Data Sources/Actuators in Airport Digital Twin Use Case

Scenario "A": Scheduling Operators Depending on Aircraft Traffic
This scenario aims to improve the schedule of turnaround tasks among the operators of an airport based on the position of aircrafts and information of flights. When a new aircraft is parked in a stand, the AirportNavigationApplication updates the 3D model of the airport, the OperatorManagementSystem assigns tasks to the operators, and the operators receive in their tablet details about their work and the place they have to go. The following data flows may occur.

Table 4.3.3-1: Scenario "A": Scheduling Operators Depending on Aircraft Traffic. Flow 1

Query/Notification from AirportNavigationApplication	Response from NGSI-LD platform	Sources of information used to answer the query
Subscribe to notifications for aircraft position update	New aircraft X present in stand Y	AircraftLocationSystem
	Aircraft X leave stand X	AircraftLocationSystem

Table 4.3.3-2: Scenario "A": Scheduling Operators Depending on Aircraft Traffic. Flow 2

Query/Notification from OperatorManagementSystem	Response from NGSI-LD platform	Sources of information used to answer the query
Subscribe to notifications for new turnaround tasks	New turnaround task of flight X with priority Y	AirportInternalInformationSystem
Query: What turnaround I have assigned?	Turnaround tasks of flights Xs with these priorities	AirportInternalInformationSystem
Notify: task completed	n/a	n/a

Scenario "B": Estimating Flight Delay
This scenario aims to register incidences on turn around events and estimate delays of flights. The following data flows may occur.

Table 4.3.3-3: Scenario "B": Estimating Flight Delay. Flow 1

Query/Notification from OperatorManagementSystem	Response from NGSI-LD platform	Sources of information used to answer the query
Notify: incident of type X in flight Y	n/a	n/a

Table 4.3.3-4: Scenario "B": Estimating Flight Delay. Flow 2

Query/Notification from AirportNavigationApplication	Response from NGSI-LD platform	Sources of information used to answer the query
Subscribe to notifications for flight status	Changes in actual, calculated, estimated, scheduled or target times of flight X takeoff/landing.	AirportInternalInformationSystem
Query: What is the time information about flight X?	Changes in actual, calculated, estimated, scheduled or target times of flight X take off/landing.	AirportInternalInformationSystem
Query: What are the incidents registered for flight X?	Registered incidents of flight X.	AirportInternalInformationSystem

4.3.4 Gaps

Figure 4.3.4-1 shows the information model of the Airport Digital Twin use case.

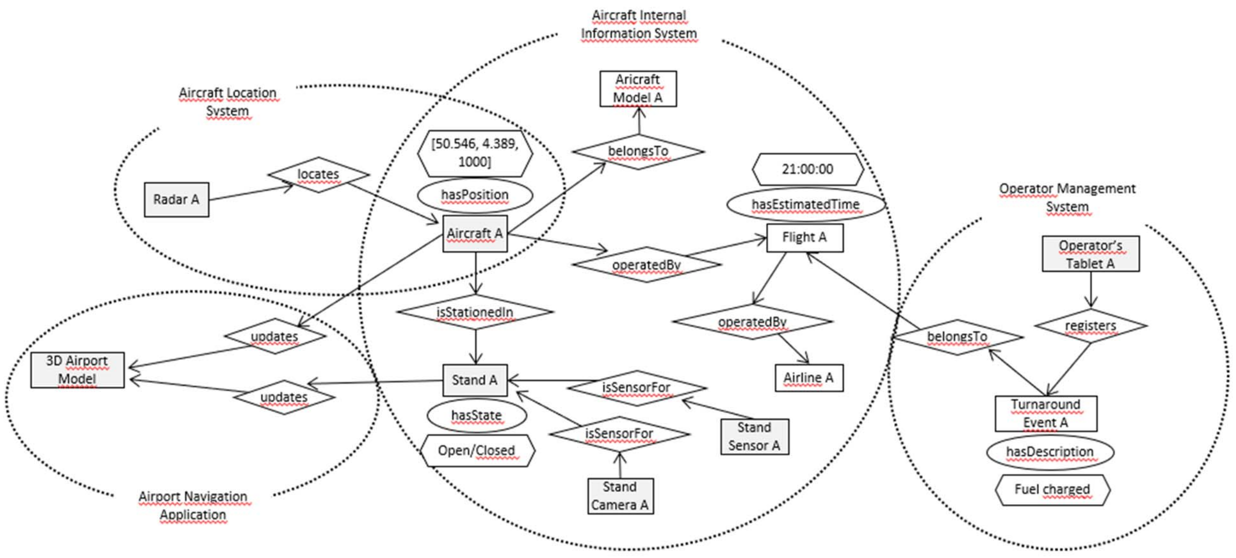


Figure 4.3.4-1: Digital Twin Use Case Entity instances graph

4.5 eHealth monitoring system

4.5.1 Usage

Healthcare is yet another area where it is clear that the concept of DT can be useful: for instance, a doctor can perform a diagnosis on patients by also studying the state of their DTs. One thing that usually doctors lack when they examine their patients is how data is evolving in time: they can only trust their patients' abilities to describe their past health status. This could be a problem when the patients themselves are not able to understand when something was wrong.

An interesting application would be an eHealth monitoring system that helps gathering patients' health parameters during they daily life. The system should be able to retrieve eHealth data from multi-modals sensors. These sensors may vary in different ways. For instance, they may be wearables (e.g. smart-bands, smart-rings, etc.) or they may be installed inside patients' house (such as thermometers).

The aggregation of eHealth classical measurements (such as heart-rate, pulse ox, sleep time) with ambient measurements (such as temperature or humidity) can allow doctors to have a better understanding of the health status of their patients: doctors would have the ability to track the values of a specific type of measurement (e.g. heart-rate) and see its evolution in time.

If required, a doctor may be allowed to change the working parameters of a sensor to modify its behaviour, in order to switch on/off of a specific group of settings, thus enabling customization based on the patient's health status.

The system may also be able to understand if there are anomalies, and it alerts both the patients and their doctors to possibly find the remedy. In this context, anomalies are both out-of-range values and "acceptable" values that the system itself considers as "critical" or "dangerous" for a patient, given his/her recent history of eHealth measurements.

In summary, the main features such a system need to comply to, would be:

- Gather data from heterogeneous eHealth sources of any type (cloud, smart sensors, etc.)
- Analyse the data in order to notify anomalies
- Send alerts whenever an anomaly has been found
- Present data in an interactive and easy-to-use dashboard
- Allow doctors to influence sensors' behaviour through their DT counterparts (actuation)

4.5.2 Challenges

The main challenges for such a system are:

- Privacy need to be preserved. Only authorized people may be allowed to access patients' data
- Data may be gathered or retrieved from source in many different ways (Cloud, Bluetooth® connections, WiFi®, etc.)
- Data of the same kind (e.g. health rate measurements) should be represented in the same way regardless of their data source

4.5.3 Solution Proposal

Domus Sapiens [i.6] is a NGSI-LD compliant system, which allows eHealth tracking and monitoring, carefully built to the most recent technologies, able to ensure specific levels of robustness, efficiency and security.

Figure 4.5.3-1 shows the architecture of the proposal. There are three main actors: **sensors**, the **middleware** and **users**.

Sensors are the data source of the system: they gather all data which is then sent to/retrieved by a specific edge node of the Middleware (the IoT Agent). This data is in raw format (so it is not NGSI-LD yet).

Sensors can either be smart or not. Smart sensors are sensors that can talk with the IoT Agent by either sending data spontaneously or when prompted for, and they can do it on their own after a simple configuration (e.g. Garmin wearables). Unfortunately, sensors are not smart, normally. For instance, usually, thermometers do not have internet connection, so they can not talk with the IoT Agent. In this case, additional hardware (e.g. Raspberry Pi) or software (e.g. mobile apps) should be provided and deployed.

The system gathers two types of data:

1) **Vital signs** of a patient, which are heart-rate, pulse-ox, sleep phases (light, rem, deep, awake) and motion intensity measurements.
2) **Ambient measurements**, which are the values of temperature and humidity (indoor and/or outdoor) taken from the house of a patient.

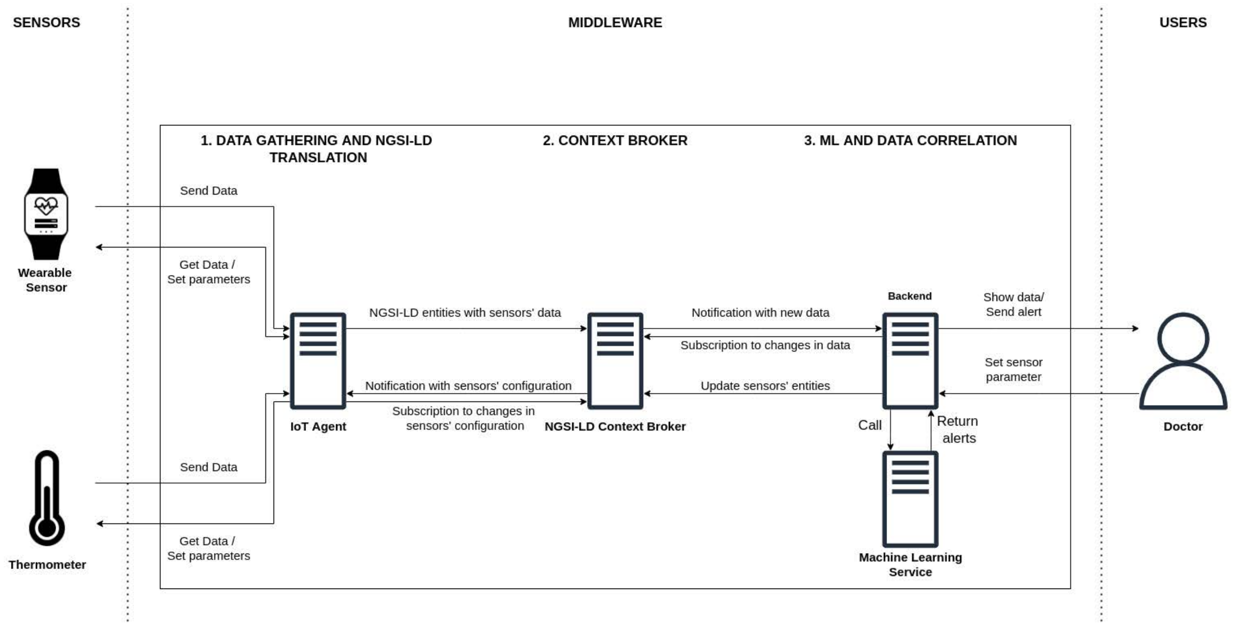

SENSORS
MIDDLEWARE
USERS
1. DATA GATHERING AND NGSI-LD TRANSLATION
2. CONTEXT BROKER
3. ML AND DATA CORRELATION
Send Data
Get Data / Set parameters
Wearable Sensor
Thermometer
NGSI-LD entities with sensors' data
Notification with sensors' configuration
Subscription to changes in sensors' configuration
IoT Agent
NGSI-LD Context Broker
Notification with new data
Subscription to changes in data
Update sensors' entities
Backend
Show data/ Send alert
Set sensor parameter
Doctor
Call
Return alerts
Machine Learning Service

Figure 4.5.3-1: Architecture of the proposed solution

The Middleware has three main sections:

1) **Data gathering and NGSI-LD translation**. This section contains the **IoT Agent** edge node. The IoT Agent gathers the raw data from sensors, translates it in NGSI-LD entities and sends them to the Context Broker. Also, it can interact directly with the sensors to change their parameters using their raw format.

2) **Context Broker**. Inside this section there is an NGSI-LD context broker instance that stores the NGSI-LD entities. It is used to notify data and to allow the actuation of sensors.

3) **ML and Data Correlation**. This section contains two elements: the **Backend** (which is the second and last edge node) and the **Machine Learning Service** (**MLS**). The backend provides a dashboard that can be used to display the eHealth data of a patient (only accessible to the patient and his/her doctors). The backend invokes the MLS every time it receives the data of a patient. The MLS processes the data and returns a list of alerts when the heart rate values are considered dangerous. If the MLS returns at least one alert, the backend sends the alerts to the users (doctors and patients) through emails and/or the mobile app.

NGSI-LD Subscriptions are used to forward data from the Context Broker in two different ways:

1) The IoT Agent is subscribed to changes in sensors' configuration.

2) The Backend is subscribed to changes in the eHealth Data.

There are two types of users: Doctors (Supervisors) and Patients. Patients can only browse their eHealth data, read the alerts generated by the system, check the status of their sensors and add doctors to access their eHealth data and alerts. Doctors can browse their patients' data and they can also change the parameters of their patients' sensors to change their behaviour.

The elements that are considered DTs are:

- Sensors
- Patients

The NGSI-LD entities that represent sensors contain the state of the real sensors. So, for instance, it is possible to know if a sensor is running or not. Also, given a sensor, it is possible to know who is its owner (if anyone owns it). Lastly, sensor entities support actuation. Actuation is triggered by modifying the value of specific attributes inside an entity of type sensor. This will trigger the Context Broker to send a notification to the IoT Agent, which will actuate the real sensor, changing its behaviour.

The DTs of patients are quite more complex. The system stores three types of information for each patient:

- Personal data. An entity of type "patients" has been created for every patient. These types of entities hold the personal data (such as the email) of every patient.
- Sensors owned. Sensor entities contain the relationship "possessedBy" which holds the URI of one entity of type "patients".
- eHealth Data. Entities that hold eHealth data of a particular patient (e.g. heart-rate data) contain the relationship "relatedTo" which holds the URI of the entity of the patient who owns that data.

The DTs of patients belong to the predictive type because the system does more than just showing the data. Specifically, the MLS is able to understand if the measured heart-rate values of a patient might suggest the existence of health problems by studying all the latest eHealth data (1-2 days) of the given patient.

An example of the information model is shown in Figure 4.5.3-2.

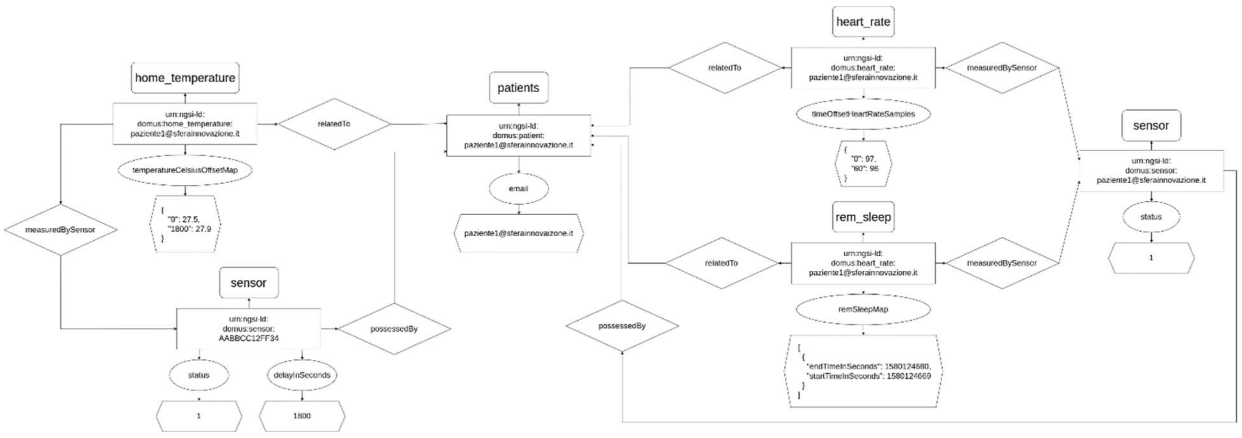

home_temperature
urn:ngsi-ld: domus:home_temperature: paziente1@sferainnovazione.it
temperatureCelsiusOffsetMap
{ "0": 27.5, "1800": 27.9 }
measuredBySensor
relatedTo
sensor
urn:ngsi-ld: domus:sensor: AABBCC12FF34
status
1
delayInSeconds
1800
possessedBy
patients
urn:ngsi-ld: domus:patient: paziente1@sferainnovazione.it
email
paziente1@sferainnovaizone.it
possessedBy
heart_rate
relatedTo
urn:ngsi-ld: domus:heart_rate: paziente1@sferainnovazione.it
measuredBySensor
timeOffsetHeartRateSamples
{ "0": 97, "60": 96 }
rem_sleep
relatedTo
urn:ngsi-ld: domus:heart_rate: paziente1@sferainnovazione.it
measuredBySensor
remSleepMap
[{ "endTimeInSeconds": 1580124680, "startTimeInSeconds": 1580124669 }]
sensor
urn:ngsi-ld: domus:sensor: paziente1@sferainnovazione.it
status
1

Figure 4.5.3-2: Partial view of the information Model

This example depicts the relationships between the entities managed by the system. The "patients" entity contains the personal data of the patient "paziente1". It is shown that a patient can own more than one sensor, and that sensors can measure more than one type of data. For instance, the sensor with id "urn:ngsi-ld:domus:sensor:paziente1@sferainnovazione.it" has measured both the heart_rate and the rem_sleep of "paziente1". Another important fact to underline is that patients may lose the ownership of a sensor, but he/she will not lose their own measurements. This has been achieved by relaying on the "relatedTo" relationships that links a measurement with the patient's entity.

As Figure 4.5.3-2 shows, the information model is "measurements centric", not "patients centric". They allow to perform different types of queries within the context broker. A "measurements centric" approach would allow to easily know the related patient from the measurement entity, while a "patients centric" approach would allow to easily query the context broker to obtain all of the measurements of a particular patient, or to obtain the list of sensors he/she owns.

For this system, the most important things are the eHealth values measured by the sensors. Indeed, the MLS needs data to perform the predictions. It does not get the patient entity as a whole, but it studies the measurements of a patient's vital parameters (such as the heart rate) whenever they are ready separately. Considering that the most common operation is the invocation of the MLS, it made more sense to choose the "measurements centric" approach.

Additionally, it would make sense to start from the patient entity - either to query all the sensors he/she owns or to query the related eHealth measurements - but this is not the case. The main reason is that the NGSI-LD information model does not support bidirectional relationships. The only way to have a bidirectional relationship would be to create two relationships to link the two entities to each other in order to navigate in both the verses. However, this solution is not practical and it could lead to inconsistency problems.

Thus, the bidirectional relationship problem has been fixed by the backend, which computes the inverse relationship on-the-fly: a needed solution to allow the dashboard to display all the data and sensors owned by a specific patient.

4.5.4 Gaps

Gaps with v1.5.x of the NGSI-LD specification are:

- There is not a standard way to perform an actuation workflow using the NGSI-LD API.
- It is not possible to define bidirectional relationships efficiently. A way would be to double the number of relationships, which is not practical and which could lead to inconsistency problems.

4.6 Elevator Use Case

4.6.1 Usage

Elevators are one of the commonly used vertical transportation systems. Digital twins for elevators are useful throughout their development, testing, and operation. Elevators' lives are long and thus their digital twins need to remain operational in parallel to the elevators. Thus, evolution of elevators together with their digital twin's evolution is important and need to be automated as much as possible. An elevator is responsible for efficient and comfortable movement of passengers across different floors in the buildings, in addition to ensuring their safety and reliability. The performance of elevators is measured with various Quality of Service (QoS) metrics such as waiting and transit times of passengers.

In particular the digital twins for elevators have the following key usage:

1) Digital twins at the design time, e.g. in both Software in the Loop (SiL) and Hardware in the Loop (HiL) simulation setup can determine the QoS of software responsible for scheduling of passengers and help optimizing it for various hardware configurations.
2) Digital twin knowledge can be exchanged across different design stages, e.g. SiL to HiL and later to the operation.
3) Digital twins during the operation can be used to optimize the performance of elevators during the operation in the presence of uncertainties.

4.6.2 Challenges

As described in the previous clause, digital twins for the elevators can be considered at two main phases, i.e. design and operation time. The key challenges related to building these digital twins are:

- During operation, obtaining data in real-time is a challenge especially due to the security reasons. Such time series data is important for updating the digital twin continuously.
- Digital twin for an elevator will remain in operation for a long period of time and needs to evolve continuously. Such evolution need to be automated as much as possible and need to be invoked in response to various changes in hardware, software, building configurations, etc.
- During design time how to simulate various aspects such as hardware realistically.

4.6.3 Solution Proposal

Our solution proposal is shown in Figure 4.6.3-1. Digital twins for elevators can be built in different settings such as SiL, HiL, and in operation. In SiL/HiL setting, relevant simulation software and hardware (for HiL) are needed. The corresponding digital twin will take time series data through context information broker and also initiate actions if needed. During operation time, obtaining data in real-time is a challenge especially due to the security reasons. Such time series data is important for updating the digital twin continuously. From a deployment perspective, there are different options to care about the security reasons isolation needed between the systems. For example, it could be hosted on the same broker with logic-based isolation. Another deployment view could be to have distributed broker with an edge broker running side by side with the elevator in operation. Given the fact that digital twin for an elevator will remain operation for long time and consequently will evolve. To this end, the NGSI-LD way to model data (knowledge graph) provides flexibility to track the environment (context) modifications. Subscription mechanisms could trigger the execution of the model recomputing anytime the context change.

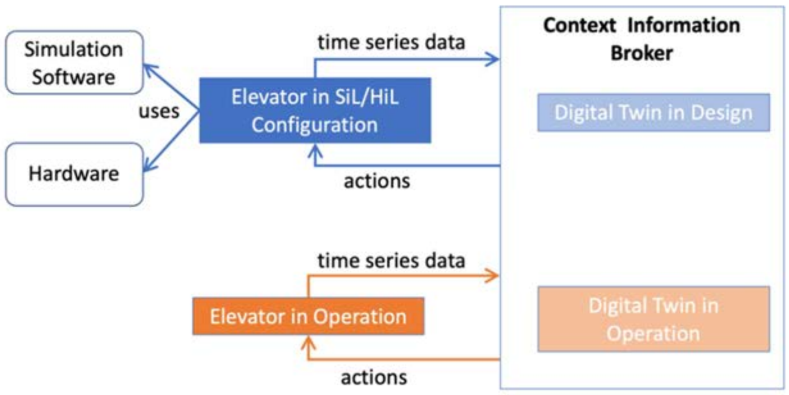


Figure 4.6.3-1: Proposed Solution for the Elevator Case

4.6.4 Gaps

NGSI-LD cannot help in realistic simulation; therefore, for design time digital twin realistic models are needed.

5 Existing initiatives and relations

This clause reviews some of the existing initiatives. It does not aim at being exhaustive and focuses on one that are relevant for the NGSI-LD context.

Table 5-1: Standardization bodies and alliances

Name	Description	Comments/Relevance
IEC TC 65 Industrial process measurement - control and automation [i.10]	IEC TC 65 - industrial process measurement - control and automation TC 65 addresses several smart manufacturing aspects, including reference architecture, smart manufacturing requirements, and asset administration shells. In the context of reference architecture, this is a specific layer on connectivity. Related to requirements, issues like AI, Edge and Cloud Computing, 5G and F5G are referenced as well in the context of new technologies. In the area of market and innovation trends, aspects like 6G, F6G, Quantum Technology, Digital Twin, digital factory and zero-touch manufacturing are addressed as well.	IEC TC 65 and ISO's TC 184 (Automation Systems and Integration) have formed a Joint Working Group (JWG21) to develop Industry 4.0 standards, and the JWG21 established the RAMI 4.0 as a major reference model of Industry 4.0 and an international standard [i.6].
Innovative Optical and Wireless Network Forum	The Innovative Optical and Wireless Network (IOWN) Forum aims to realize a smarter world where data, activities and people in different industries will be brought together in order to create a fully connected and intelligent society. The Forum is working on optical/wireless networking which can provide advanced capabilities of low-power consumption, ultra-wide bandwidth, large-scale simulations, and ultra-realistic UI/UX.	May not look relevant on first sight, but they have Digital Twins has one of their strong use cases.
ISO/TC 184 Automation Systems and Integration	ISO/TC 184 Automation Systems and Integration handles standardization in the field of automation systems and their integration for design, sourcing, manufacturing, production and delivery, support, maintenance and disposal of products and their associated services. Areas of standardization include information systems, automation, control systems and integration technologies, Digital Twins. ISO TC 184/SC 4 develops several standards including ISO 23247 (Digital Twin manufacturing	ISO 23247-3 [i.13] defines the information for digitally representing manufacturing elements, including products, processes, and resources. ISO 23247-4 [i.14] defines a method for exchanging information in a Digital Twin. The type of manufacturing supported by an implementation of the ISO 23247 framework depends on the standards and technologies available to model the observable manufacturing elements.

Name	Description	Comments/Relevance
	framework).The scopes of the four parts of this series are defined below: • ISO 23247-1 [i.11]: General principles and requirements for developing Digital Twins in manufacturing; • ISO 23247-2 [i.12]: Reference architecture with functional views; • ISO 23247-3 [i.13]: List of basic information attributes for the observable manufacturing elements; • ISO 23247-4 [i.14]: Technical requirements for information exchange between entities within the reference architecture.	Different manufacturing domains can use different data standards. As a framework, this serie of ISO documents do not prescribe specific data formats and communication protocols. This is a positioning point for NGSI-LD.
DTS/SmartM2M-103828 [i.15] SAREF: Ontology Support for Digital Twins and usage guidelines	Technical Specification on "SAREF: Ontology Support for Digital Twins and usage guidelines" to fill the priority gaps related to SAREF and will include guidelines about how to use SAREF in the context of Digital Twins and with Digital Twins	This is work in progress in STF 641 (funded by EC/AFTA) of ETSI TC SmartM2M No draft document has yet been produced. A SAREF ontology support for Digital Twins may be relevant for IoT related use cases connected through a oneM2M layer.
ISO/IEC JTC1/SC41 WG Internet of things and digital twin	The scope of ISO/IEC JTC1/SC41 is standardization in the area of Internet of Things and Digital Twin, including their related technologies with the purpose to: serve as the focus and proponent for JTC 1's standardization programme on the Internet of Things and Digital Twin, including their related technologies. Provide guidance to JTC 1, IEC, ISO and other entities developing Internet of Things and Digital Twin related applications.	The initial document (ISO/IEC 30173 [i.16]) is under development and plans to propose terms and definition relevant for Digital Twins. It however only focuses on IoT systems.
ETSI ISG ARF Augmented Reality Framework	The purpose of the ISG ARF is to define a framework for the interoperability of Augmented Reality (AR) components, systems and services. It defines an overall functional reference architecture, identifying key components and interfaces for an AR solution. The AR framework will allow AR components from different providers to interoperate through the defined interfaces. This will in turn limit vertical siloes and market fragmentation and enable players in the ecosystem to offer part(s) of an overall AR solution.	ISG ARF has an interest on open-source solutions and needs to build a contextualised representation of a viewer environment. ISG CIM defines how to handle contextualised information but does not investigate the visualization options.
W3C Thing Description Web of Things (WoT)	The Web of Things (WoT) seeks to counter the fragmentation of the IoT by using and extending existing, standardized Web technologies. By providing standardized metadata and other re-usable technological building blocks, W3C WoT enables easy integration across IoT platforms and application domains.	This one focuses on the IoT layer but included Digital Twin use cases in its group note on "use cases and requirements"

Name	Description	Comments/Relevance
IEC Technology report City information modelling and urban digital twins [i.17]	City Information Modelling (CIM) and Urban Digital Twins (UDTs) are two sets of technology-enabled practices used to help city planners develop smart cities and city officials and business leaders to manage them. Aims of the IEC technology report are: • Gain a better understanding of the state of the art of theory, technology, practice, and standards of CIM and UDT • Identify the similarities and differences between CIM and UDT • Identify the requirements for standards needed for CIM and UDT • Promote the exchange of information and experiences and enhance the conversation between CIM and UDT experts and practitioners	Several initiatives of urban digital models based on NGSI-LD are emerging in the FIWARE ecosystem.
Digital Twin consortium	The Digital Twin Consortium has three primary objectives: • Influence the direction of the Digital Twin industry • improve interoperability of Digital Twin technologies, and • influence the requirements for Digital Twin standards. • Several working groups being domains focused (i.e. energy, etc.) + horizontal one such as 3Ts (Technology, Taxonomy and Terminology) have been created to discuss building blocks for DT.	An open-source initiative has been created to host relevant open-source projects. Some Digital twin consortium members are also ETSI ISG CIM members and NGSI-LD based contributions are under discussions
Industrial internet consortium	The Industry IoT Consortium (IIC) is a global not-for-profit partnership of industry, government, and academia. Since its founding in 2014, the IIC has helped build a technical foundation for the Industrial IoT. The IIC was built to support technology and business leaders in industries that leverage IoT, including but not limited to energy, sustainability, healthcare, manufacturing, supply chain, mining, retail, smart cities, and transportation. The IIC's role is to identify and facilitate use cases and innovative enablers, deployments, and approaches that alleviate the pain felt by technology end-users.	IIC has co-authored a white paper [i.6] on the use of Asset Administration Shell concepts for Digital Twins in the Industrie 4.0 context

Table 5-2: Supporting initiatives

Name	Description	Comments/Relevance
Digital Twin Hub Community	The mission of the Digital Twin Hub Community is to create connections and foster collaboration between Digital Twin owners and information management experts enabling the UK to realize the potential of connected Digital Twins.	Several of the use cases presented at the OGC online event in January 2021 "Urban Digital Twins" were funded under that initiative.
Destination Earth	The objective of the Destination Earth initiative is to develop a very high precision digital model of the Earth to monitor and simulate natural and human activity, and to develop and test scenarios that would enable more sustainable development and support European environmental policies. DestinE will be implemented gradually over the next 7-10 years, starting in 2021. The operational core platform, the Digital Twins and services are scheduled to be developed as part of the Commission's Digital Europe programme, whilst Horizon Europe will provide research and innovation opportunities that will support the further development of DestinE. Synergies with other EU programmes, such as the Space Programme, and related national initiatives are also explored.	This is being implemented by the European Space Agency together with 2 other agencies related to weather observation and satellite exploitation.

Table 5-3: Open-source initiatives and related technologies

Name	Description	Comments/Relevance
FIWARE foundation	FIWARE Foundation is a non-profit organization that drives the definition and encourages the adoption of open standards (implemented using Open Source technologies) that ease the development of smart solutions across domains such as Smart Cities, Smart Energy, Smart AgriFood and Smart Industry, based on FIWARE technology	FIWARE ecosystem is the primary supporter and contributor of the NGSI-LD specification by providing compliant open-source enablers. Digital Twin are in the core of the FIWARE Foundation strategy
Microsoft DTDL	Microsoft is proposing a Digital Twin Description Language (DTDL) based on JSON-LD which is programming-language independent. DTDL is made up of a set of metamodel classes that are used to define the behaviour of all Digital Twins (including devices). There are six metamodel classes that describe these behaviours: Interface, Telemetry, Property, Command, Relationship, and Component.	In cooperation with Open and Agile Smart Cities (OASC), a mapping of DTDL to ETSI NGSI-LD has been made for the smart city context
Eclipse Ditto	Eclipse Ditto is an open source framework helping to build digital twins of devices connected to the internet. It is by design domain agnostic and thus may be used in industrial, residential, agricultural and many other IoT domains. It abstracts the device into a digital twin providing synchronous and asynchronous APIs and use the digital twin API to work with the physical device.	Focuses on Things level and could be use on the ingestion layer to provide device management functions
mago3D	mago3D is a 3D platform that integrates management and visualization of the AEC (Architecture, Engineering, Construction) area and traditional 3D spatial information (3D GIS).	It is building information management (BIM) oriented and focus on accelerating complex geographical visualization in the web browser
GAZEBO	GAZEBO offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. It includes a physics engine, high-quality graphics, and programmatic and graphical interfaces. Gazebo is free with a community support.	Some platforms use gazebo as a backend to allow web based representation It is the de facto simulator for all people in the robotic field.
Asset Administration Shell Explorer	AASX Package Explorer is a C# based viewer/editor for the Asset Administration Shell.	
PLATEAU	PLATEAU is a project for 3D city model development promoted by the Ministry of Land, Infrastructure, Transport and Tourism of Japan. It uses the OGC CityGML 2.0 for 3D representation and VCDatabase/3D City database for 3D data storage.	Connecting with n3D City models (e.g. CityGML) is a field to explore with NGSI-LD

6 Purpose and usage of Digital Twins

6.1 Classification of Digital Twins

6.1.1 Definition

In NGSI-LD a Digital Twin is an entity which can be atomic or be the entry-point of a graph. Each Digital Twin:

- is universally identified with a URI (Universal Resource Identifier);
- belongs to a well-known type also universally identified by a URI; and
- is characterized by several attributes which in turn are classified as:
  - properties holding data; or

- relationships, each targeting another Digital Twin entity identified by a URI.

The Digital Twin is the digital counter part of a real-asset, which can be a physical asset or a concept.

EXAMPLE: With NGSI-LD, Digital Twins can then represent the wheel of a bus, the bus, the city public transportation system or the bus ticket.

Three levels of Digital Twin complexity can be represented within an NGSI-LD property graph and are described hereafter.

6.1.2 Atomic-twins

Most assets management platforms (such as IoT platforms) do already maintain some kind of very minimal Digital Twin (understood as a mere proxy) for the assets they take charge of. The term should not be used by default, just for riding a hype wave. These "atomic-twins" make sense as the lowest rung of a ladder, nested within higher-level and larger-scale graph-based-twins of the environments they fit into. These "atomic-twins are represented directly as NGSI-LD entities (graph vertices) because they need not or cannot be decomposed further internally into sub-systems, which means they are viewed as black boxes. They will have an assortment of static and dynamic attributes of their own, possibly capturing part of their state in the sense of dynamical system theory, which is already much more than what current asset management platforms offer. In case of connected devices, these proxies may provide a network interface supporting direct interaction (e.g. actuation) of applications with the device.

6.1.3 System-Twins (STs): self-contained systems as rooted subgraphs

Well-defined subgraphs of the overall graph will capture the key structural links that make up the scaffolding of a "classical" self-contained system, in the intuitive sense of a physically-enclosed appliance, stacking together a set of parts/subsystems which are its direct constituents. These constituent subsystems may either be captured as "atomic-twin" atomic vertices as described before, or decomposed further, recursively, into subsystems which may themselves be described by the same kind of rooted subgraphs.

Figure 6.1.3-1 gives an example of this for an apartment captured as a subsystem of a building, decomposed further into rooms, which are themselves composed of entities considered here as atomic and captured as "atomic-twins".

These subgraphs have a "root" node standing for the subsystem being described by the subgraph (building, apartment, room), but the overall connection pattern is not limited to a directed rooted tree (also known as arborescence). Typically they will "look like" undirected rooted trees, with added transversal links, mostly undirected edges, between the "vertical" branches that lead to the root, as shown in Figure 6.1.3-1 (with NGSI-LD relationships drawn as diamonds and entities as rectangles).

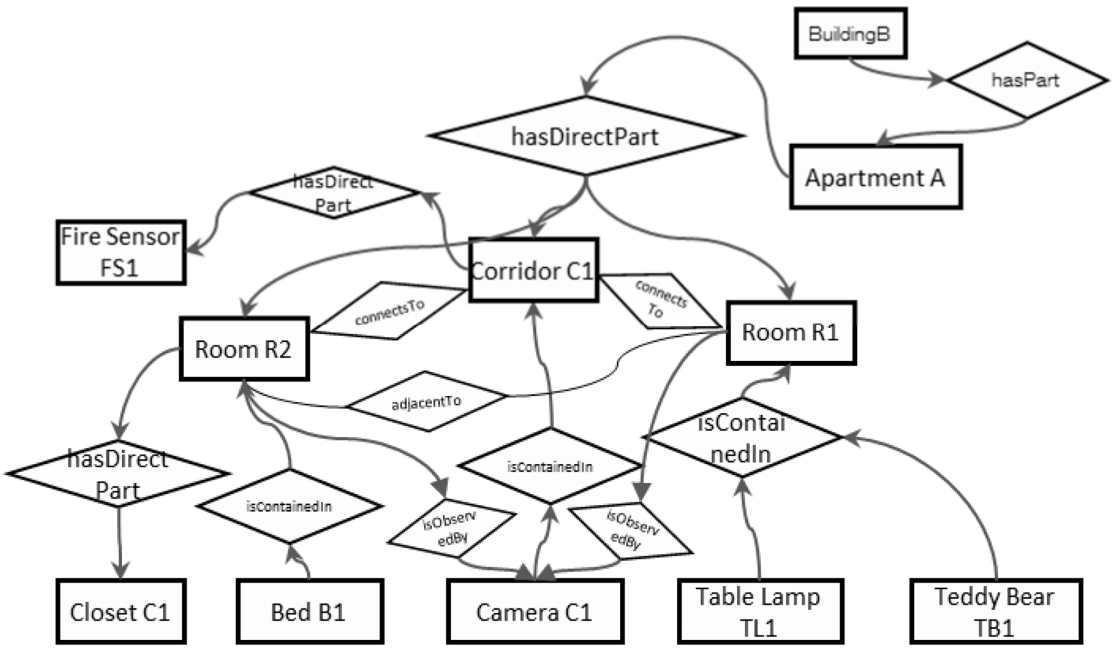


Figure 6.1.3-1: Description of a building as self-contained system, with nesting of subsystems

By definition, the relationships between the nodes that make up self-contained systems like these are physically local; they may correspond to:

1) Vertical *top-down* links between the root system and its constituent parts (e.g. with type *NGSI-LD:hasPart/hasDirectPart* as recommended in [i.8]), when these parts are designed to be included in the overall system and this system cannot work if these parts are removed.

2) Vertical *bottom-up* links (e.g. with type *NGSI-LD:isContainedIn* as recommended in [i.8]) between parts and systems that would not always be captured as such, like the sets of all things (furniture, appliances, etc.) contained inside a room. This type of relationship may also be used to capture an even more informal and purely contingent set-based location, without implying a "systemic" relationship.

3) Transversal links (e.g. with type *NGSI-LD:ConnectsTo* as recommended in [i.8]) to capture the way through which one may go from one room to another, or a room is near another (e.g. with type NGSI-LD:AdjacentTo as recommended in [i.8]).

4) Transversal directed links between a sensor and what it observes (e.g. with type *sosa:isObservedBy*), or an actuator and what it acts upon (e.g. with type *sosa:isActedOnBy*).

As seen in the examples from Figure 6.1.3-1:

1) there is at least one path leading from the root to any node in these subgraphs.

2) there may exist cycles in these subgraphs.

3) the root is special only in that it is never:

   - the source of a "isContainedIn" relationship with a target in the same subgraph;

   - the target of a "hasPart" relationship with a source in the same subgraph.

   The last two features distinguish them from both graph-theoretic directed-rooted-trees and undirected trees.

It is reminded that the NGSI-LD specification only support directed relationships. When required, inverse direction of relationships such as "isContainedIn" or "hasPart" should then be formally created using relationships such as "contains" or "isPartOf".

6.1.4 Systems of Systems Twins (SoSTs): capturing distributed and complex systems

This further level of representation does also capture systems as subgraphs of an overall reference graph. It is used for a less obvious, but critically important type of systems that are labelled here "Systems of Systems" (SoS) for short, even if not all of them, by far, are SoS in a strict sense [i.3]. For these more complex systems, it would make no sense, or be impractical, to have a regular direct relationship between a "root" node and the constituent parts of the systems, as proposed for the simple self-contained systems described before. The reasons why this kind of system has to be captured in this special way may be one or several (*but not all*) of the following:

1) the system is physically distributed, potentially on a very large scale, with a large number of direct constituents;

2) the system belongs to the broad category of *physical infrastructure networks*, whose connections correspond to actual physical links: road/street networks, electrical grids, water distribution networks, gas distribution and transport networks, telecom networks (at the physical infrastructure level), etc.;

3) the system is a "system of systems" in the strict sense of the customary definition: a bottom-up assemblage of subsystems which are *operationally independent* and *have not been designed to work together*, but which do happen to *work together to provide a functionality that is more than the sum of those provided by the individual subsystems separately*; like the Internet at large, or a city, be it smart or not! ;

4) the system exists mostly as an informational abstraction, grouping subsystems that are physically independent, or have a loose connection, like e.g. a logistical network, a waste collection network, or, in a different vein, a sharing system that federates a set of physical assets for rental or lending;

5) the relationships between the parent system and its constituent subsystems correspond neither to the "*NGSI-LD:hasPart*" relationship characteristic of a classically engineered top-down system, nor the more informal "*NGSI-LD:isContainedIn*" relationship of a bottom-up, "informal" system.

These subgraphs are clusters of nodes, together with the relationships that bind them and the properties that characterize them. They are impersonated by higher-level "hypernodes" with type "*NGSI-LD:graph*". The relationship between constituent nodes and their parent hypernode are captured by a special "*NGSI-LD:isNodeOfGraph*" relationship. The graph "hypernodes" may themselves be nodes in a higher-level graph, as subgraphs of this graph, with a relationship of type "*NGSI-LD:isSubGraphOf*".

An example of this type of systems-of-systems grouping is illustrated below for the infrastructure of a smart city, with obviously a tiny sample of the kinds of nodes that each of these systems would comprise.

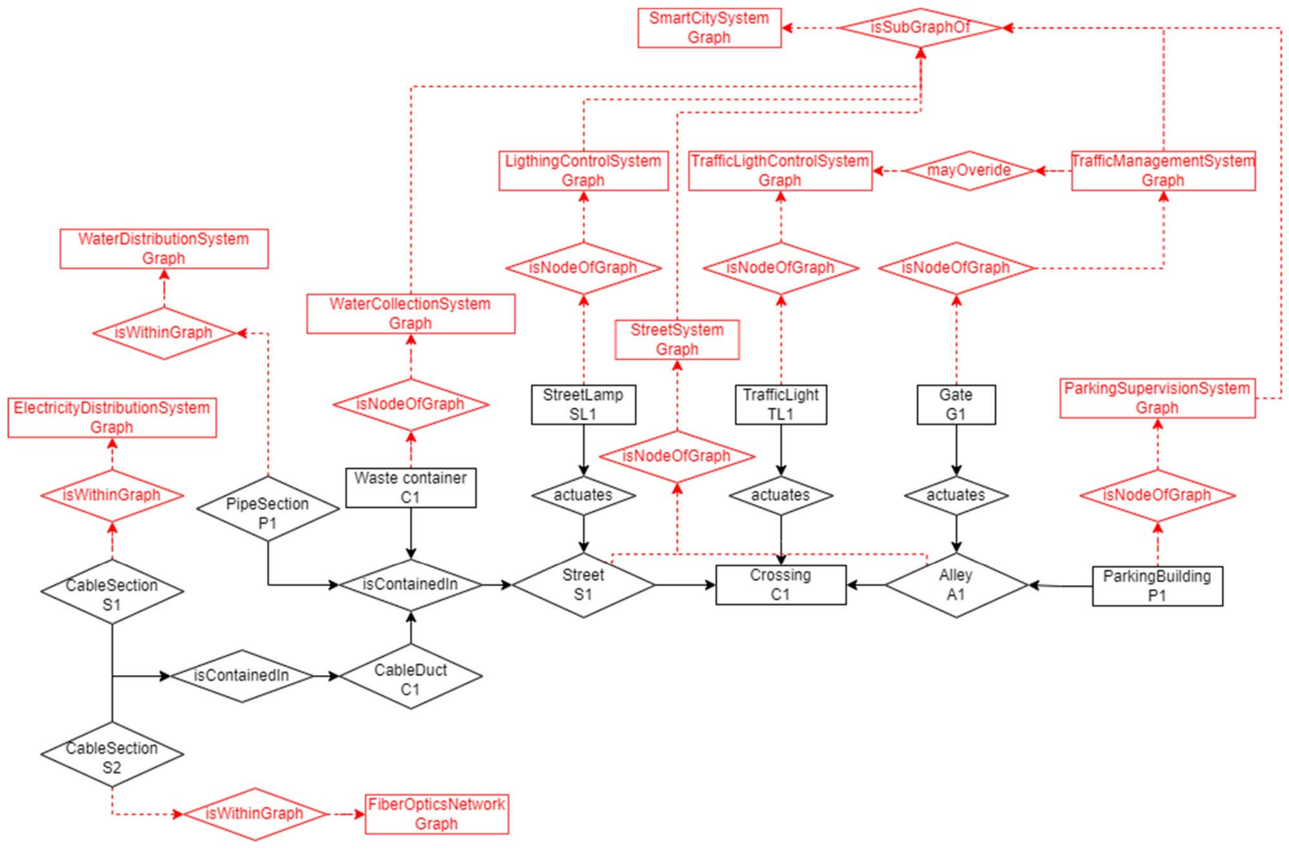


Figure 6.1.4-1: Subsystems of city infrastructure, captured as separate subgraphs and represented by graph "hypernodes"

As a matter of fact a "graph" node *stands for a whole subgraph, which is itself a system twin*, but such a graph node is not at all equivalent to the root node of a rooted graph.

Only the simplest types of self-contained top-down classical systems can be represented by rooted subgraphs, without using this more heavyweight graph construct, which does also require "isNodeOfGraph" relationships between the nodes of the subgraph.

6.2 Digital Twin capabilities

6.2.0 Introduction

Digital twins intensively rely on data that they consume and produce with the purpose of providing the user with a digitalised, augmented and actionable view of the real-world asset.

6.2.1 Descriptive twin

The descriptive twin informs about the current state of the real-world asset. The descriptive twin presents past and current values of some of the real-world asset characteristics. These characteristics can be static (e.g. a building geometry) or dynamic (e.g. sensor measurements). The connection between the real-world asset and the Digital Twin is bidirectional: a change made in the descriptive twin is reflected (in real-time or not) on the real world asset (actuation). The descriptive twin capability does not restrict to metrics collected from the real-world asset and can be augmented with computed functions (e.g. distance to a fix point of a moving real world asset; deviation from normal of a room temperature).

Descriptive twins are natively handled by any NGSI-LD compliant broker. Some augmentation functions (e.g. time series aggregation functions) are also natively included in the NGSI-LD specification.

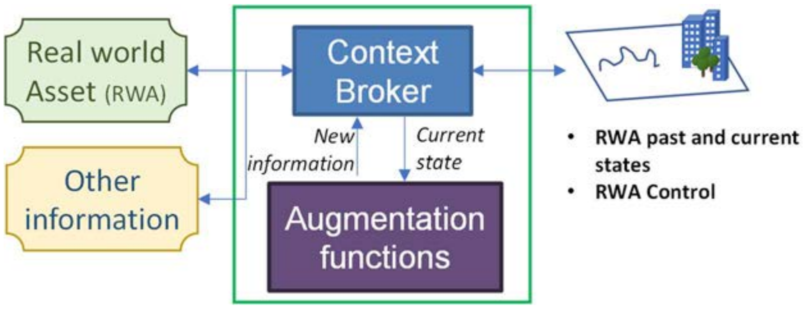


Figure 6.2.1-1: Descriptive twin model

EXAMPLE: The topology of a water distribution network equipped with sensors is described as an NGSI-LD graph. Status (e.g. water level, open/close, flow, etc.) of the real assets composing the water network (e.g. tanks, valves, pipes, etc.) are available as properties of their Digital Twins being entities of the NGSI-LD graph.

6.2.2 Predictive twin

The predictive twin extends the descriptive twin capability by providing predictions on the way the real asset could evolve in the future. The predicted future values can be computed on demand or stored within properties of the Digital Twin entity(ies) to be consumed as any temporal value through the NGSI-LD temporal API. Any past, current and future values of the real asset can then be consumed by external applications using the same NGSI-LD API.

The predictive twin capability is handled by an NGSI-LD broker coupled with a prediction model which may require the availability of additional context information.

Predictions are not ensured to be completely accurate but informative about possible futures. Different possibilities (e.g. use of multi-attribute support) exist within the NGSI-LD specification to separate the Digital Twin properties values based on their source or based on other meta-information such as a confidence level associated with a prediction.

EXAMPLE: Based on the current state of a water distribution network, a water distribution simulator (e.g. EPANET) can predict the forthcoming states of the water network (pipe pressures and water flow, water tank levels, etc.) using additional information, such as the foreseen water demand curve.

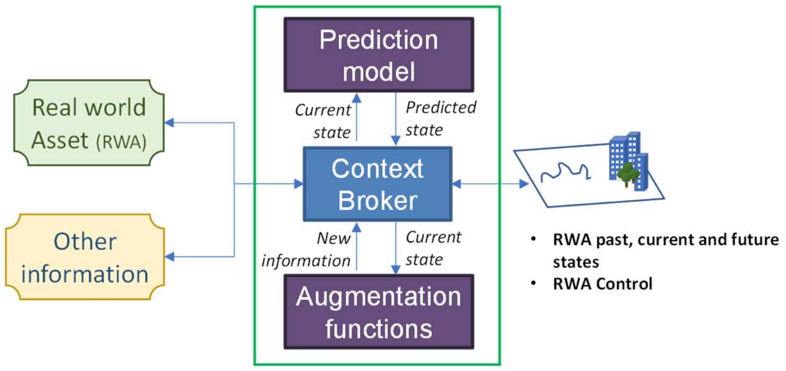


Figure 6.2.2-1: Predictive twin model

6.2.3 Prospective twin

The prospective twin builds upon the predictive capabilities. It is meant to help the Digital Twin's user to evaluate the impact that actions would have on the real-asset, if applied. This allows running "what-if" analysis made by executing the prediction model over the current description of the asset, which is altered to represent the intended action.

EXAMPLE: In the case of a water distribution network, the prospective twin allows evaluating the impact of closing a valve, considering the current status of the water network and foreseen environmental context variations.

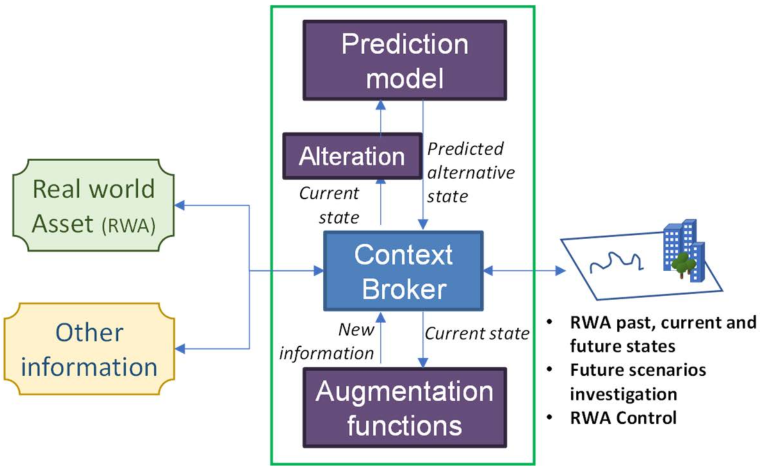


Figure 6.2.3-1: Prospective twin model

6.2.4 Prescriptive twin

The prescriptive twin goes one step further: it aims at identifying actions to be taken over the real asset for it to reach a target state. Identified actions could be simply provided to the user or directly actuated on the real asset, thus implementing an autonomous behaviour of the asset.

EXAMPLE: The water distribution network has to limit the water pressure in the pipes depending on their characteristics (e.g. nature, age, etc.) while fulfilling the demand curves at water outlets. This multi-criteria system optimization uses the prospective twin to explore options and provide an optimum configuration of water valves.

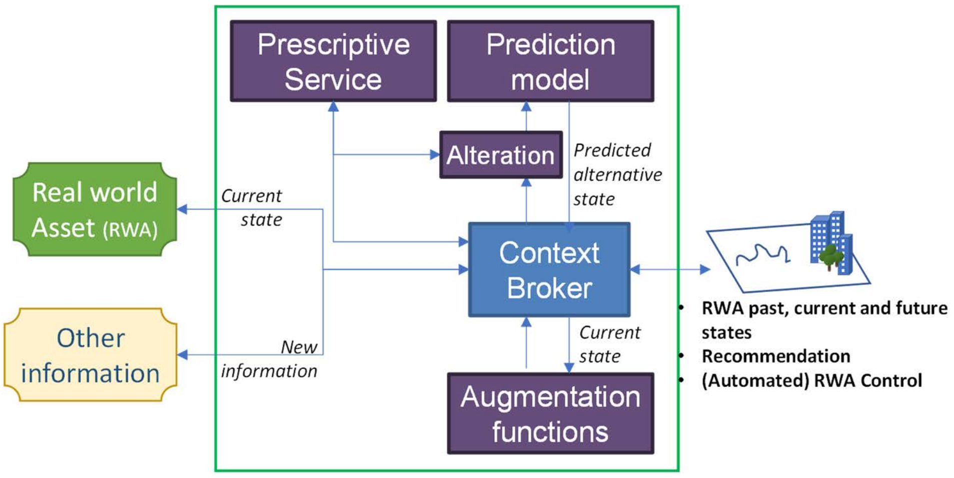


Figure 6.2.4-1: Prescriptive twin model

6.2.5 Diagnosis twin

The diagnosis twin is a particular capability which is meant to understand what happened in case of real asset malfunction or what is happening in case of a real asset deviating from its expected working trajectory. It builds on a diagnostic model that collects additional data related to the real asset evolution (e.g. log files).

EXAMPLE: The energy consumption of a water pump increases to maintain a level of water pressure at the outlet. The analysis of maintenance logs shows that the cleaning of the filters at the inlet is overdue.

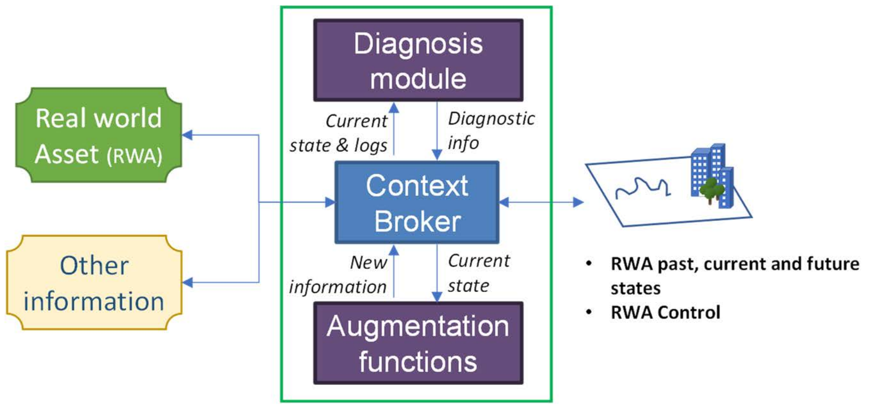


Figure 6.2.5-1: diagnosis twin model

6.3 Digital Twin lifecycle

Here is an attempt to identify what the expected functionalities and usages are. It is proposed to split the analysis over the different stages of the real asset lifecycle, including the design, the commissioning, the operation and the decommissioning stages:

- **Design stage:** A Digital Twin can be used to design a physical twin. In this case, the Digital Twin precedes the existence of the physical twin. It models the important aspects of the Digital Twin. This can include a 3D model that can be integrated in a 3D environment together with the 3D models of other twins, e.g. when building a smart factory. It can also build upon the predictive/prospective capabilities of an existing Digital Twin. This enables checking whether everything fits together as expected. This can also include physical aspects, e.g. the characteristics of an antenna and the interaction with the environment. The Digital Twin created at design stage can evolve to fulfil the requirements needed at later stages, including the relevant models needed for simulation, the history including the production of the physical twin and the different phases of commissioning, operation and decommissioning.

- **Commissioning:** At the commissioning stage, a Digital Twin can be used for configuring and putting a physical twin into service. The initialization information (such as initial configuration parameters and status of the real asset), is persisted in the Digital Twin and plays an important role for maintenance and analysis, enabling the improvement of the overall operation of the physical twin including the interactions with its environment and other Digital Twins.

- **Operation:** In the operation stage, the Digital Twin provides complementing functionality to the physical twin, which can include digital models of relevant aspects, logic with which the operation of the physical twin can be influenced or even controlled and the history of the physical twin. In particular. the following features can be provided building upon the capabilities defined in clause 6.2:

    - The Digital Twin can provide an interface to the physical twin, which can be a composed system, exposing an "equivalent system" at hypernode level.

    - Based on the available model, the current state, the history and/or information about the environment and from other Digital Twins, the Digital Twin can do predictive analytics, i.e. provide a prediction of the system state in the future.

    - The Digital Twin can monitor if the system deviates from its anticipated state following an underlying model, e.g. a state machine, to detect outliers and anomalies.

  - The Digital Twin can simulate what-if scenarios, taking into account the current state and different possible changes. Depending on the results, actions on the physical twin can be taken, influencing or controlling its behaviour.
  - The Digital Twin can optimize the operation of the system, e.g. by analysing the history, using information from other Digital Twins and doing simulations. The determined optimized settings can then be propagated to the physical twin through actuation.

- **Decommissioning:** At the decommissioning stage, the physical twin is put out of service and is possibly disassembled and recycled. The important information about it, including its history of design and operation can be persisted in the Digital Twin, which allows gaining insights for future generations of physical twins long after the original physical twin has ceased to exist.

6.4 NGSI-LD architecture for Digital Twin systems

A typical representation of a NGSI-LD architecture relevant to handle Digital Twin capabilities described earlier is provided in Figure 6.4-1. Several components have been identified and build together the Digital Twin system:

- The NGSI-LD context broker is the central component and complies with [i.8] It contains the graph instance of the Digital Twin expressed as Atomic Twin (see clause 6.1.1), System Twin (see clause 6.1.2) or System of System Twin (see clause 6.1.3).
- The connection with the real-world asset is handled by System Adapters. These are in charge of interfacing the NGSI-LD API with the interface of the RWA. Complexity of system adapters can vary. Simplest one would publish information from an external system into the context broker, supporting the Descriptive Twin capabilities whereas more complex System Adapters would be able to handle complex actuation logic as proposed in clause 7 in a completely secured way.
- Additional services are optionally deployed to realize the Digital Twin capabilities and interface with the context broker through the NGSI-LD interface:
  - **Augmentation functions** provide additional descriptive information in the Digital Twin attribute based on the observed state as well as other information which could be made available to the context broker. These could build upon NGSI-LD embedded capabilities such as aggregation function (e.g. to provide a moving averaging of a time serie) or an external service (e.g. adding engine characteristics of a car using its licence plate information to consult a national database of car registrations).
  - **Prediction models** are consuming information about the Real World asset and possibly other information to provide prediction. They can be executed on-demand or triggered when new information about the Digital Twin is made available (e.g. using the NGSI-LD subscription mechanisms). No assumption is made on the type of prediction model which can be a physical model, machine learning based, etc. The description of the current state of the Digital Twin could be altered to run predictions over a hypothetically modified version of the Digital Twin so to execute the prospective analysis.
  - **A Prescriptive service** can be deployed to provide recommendations related to the Digital Twin managements. As an illustration, this service can interact with the prediction model, testing different alterations, to identify an optimum functioning point of the Real World asset.

The communication flow across these services and triggering of the corresponding processes can be managed using a pure data model driven approach, building upon the NGSI-LD API, as experimented in the use case described in clause 4.1. However, more complex flows could be orchestrated using new approaches to handle service execution as described in clause 7.3.

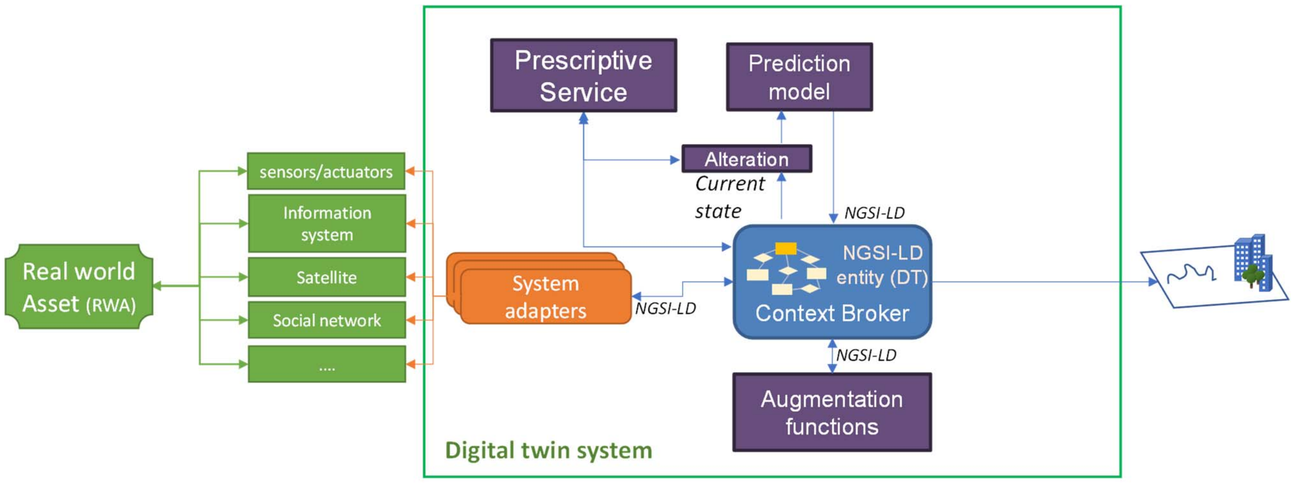


Figure 6.4-1: High-level architecture for Digital Twin systems

7 Proposed orientations for NGSI-LD specification

7.1 Summary of identified gaps and recommendations

The use cases provided in clause 4 highlighted the potential of NGSI-LD specification to represent and manage digital twins in even complex scenarios as described in clause 6. To improve the developers and users' experience, an analysis of the possible evolutions to be considered in forthcoming updates of the specifications:

- **Handling of actuation and services execution**: several initiatives have demonstrated the use of the NGSI-LD specification to handle actuation. This deserves further specification work to handle complex scenario of Digital Twin service execution.
- **Handling of simulations**: the deployment and management of simulators is left as an implementer choice. The capabilities and architectural descriptions made in clause 6 clarify the global approach which is relevant to handle digital twins.
- **Handling of the graph structure**: the existing API specification as well as architectural choices (capacity to handle distributed and federated scenarios) limit the possibilities to make multi-level graph traversal queries or graph operation in general. The need to make the NGSI-LD relationships first class citizen in the API, as the NGSI-LD entities are, is also wished.
- **Handling of 3D spatial data**: There is a tendency for cities to develop their 3D representation model and need to set direct connection with the sensors and actuators deployed in the city are emerging, to build the 3D Descriptive twin of the city. While NGSI-LD does not intend to handle 3D representations by itself, its linked data representation allow linking to external 3D models.

The two later points are topics which requires further investigations to clarify the positioning of the NGSI-LD specification within the knowledge graphs area as well as 3D representations. The work being pursued in the context of the specialist task force STF627 in the year 2022 is going to pave the way for further discussions.

The topic of improved specification related to the handling of service execution is described further in that clause.

7.2 Suggested Actuation Workflows for Digital Twins using current NGSI-LD API

7.2.1 Actuators and feedback to the application

Actuators are assets that can change their state (light on/off) or execute actions (move forward, detect face, etc.). The application interacts with an actuator through the actuator's Digital Twin System.

There is currently no explicitly and precisely specified support for actuation in the NGSI-LD API. Thus, this clause describes some conventions that represent a proposed best-practice about how NGSI-LD API and data models can be used for the interaction between applications and actuators represented by NGSI-LD-based Digital Twin Systems.

The conventions and approach described in this clause are not powerful enough to implement complex actuation jobs that depend on each other and, for instance, make actuation decisions conditional on the outcome of other actuations, unless that behaviour is implemented in a custom way within the application logic. Thus, the concept of a more evolved Service execution logic, being a first-class citizen of the NGSI-LD API and able to offer more structured building blocks for actuation, is described in clause 7.3.

A Digital Twin System that comprises an actuator and supports actuation workflows is represented as one or more NGSI-LD Entities and System Adapters, which collaborate.

The application-actuator interaction needs to be bidirectional. Thus, actuators are triggered by the reception of actuation-specific commands (e.g. "set the on state of the lamp to false", "send a SMS") that are encoded as NGSI-LD data, following a suggested data model. They respond with feedback, similarly encoded as NGSI-LD data.

Command feedbacks may serve to control the maximum operations rate a controlling application needs to achieve, and different levels of feedback can be requested, by associating a specific Quality of Service value to the command:

- Some applications need high operation rate but no feedback. For this case a QoS = 0 can be used. The typical example is to control the arms of a robot with a joystick.
- Some applications need to be sure that the actuators actually received the command request or need to get back a payload in response to the command. In this case a QoS = 1 can be used. The typical case is switching on a light with confirmation, or request face-detection with consequent notification of matching events.
- Commands can either require a short or a long execution time. For commands with long execution time, the application may require a continuous status feedback. In this case a QoS = 2 can be used. The typical example is that of a door opening, where feedback continuously report the current level (10 % open, 50 % open, …).

7.2.2 Architecture for actuation

Commands are sent to the Digital Twin System by the application, using the standard NGSI-LD API and a suggested convention for representing them. In turn, feedback about command execution is received by the application, both as continuous status updates and/or a final command result.

More specifically, within the Digital Twin System, the component that handles direct communication with the actuator is the System Adapter: it uses an actuator-specific protocol to control the actuator and get responses and updates from it, i.e. from the real system. On the other hand, the System Adapter is able to use the NGSI-LD API to receive NGSI-LD command requests from the NGSI-LD Entity and send back command status and result to it. The NGSI-LD Entity is responsible for handling direct communication with the application.

Thus, to support actuation, there is a need to specify:

- Additional NGSI-LD Properties the Digital Twin System need to have, in order to represent and manage command Request, Status, Result.
- A communication model that allows commands to flow in forward direction and feedback to flow in reverse. This communication model need to comprise a mapping, to be held within the Digital Twin System, that is able to route the command requests to the appropriate handler within the System Adapter and vice-versa.

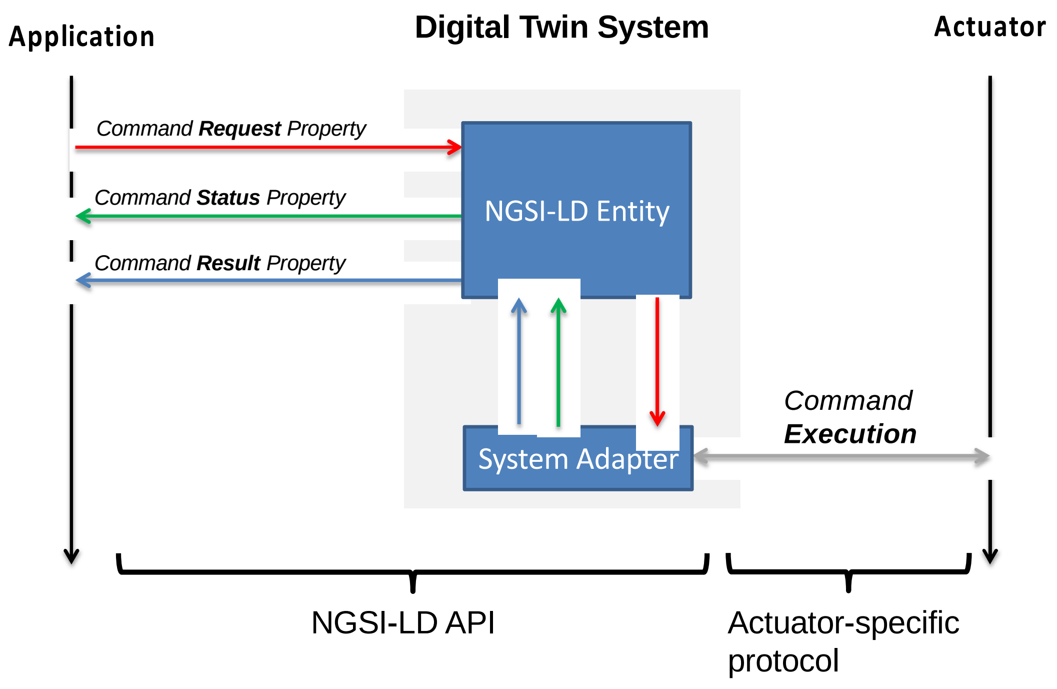


Figure 7.2.2-1: Architecture for actuation

7.2.3 Structure of Commands and additional Properties

7.2.3.0 Introduction

The Digital Twin System need to have, in addition to the usual NGSI-LD Properties representing the Digital Twin status, a set of additional, dedicated NGSI-LD Properties associated with:

- the list of available commands, i.e. the list of commands supported by the actuator;
- command endpoints, one for each command, that are used to send and receive command related messages and optionally hold state for the ongoing commands.

The structure of the commands needs to be specified, but not the internal format of their payloads. By using commands with a custom payload, one can support all kinds of operations, for example:

- "set-on": "true"
- "detect-face": {"face-features": "…."}
- "move": "forward"

The data model for command requests, status and responses has to include metadata such as the QoS of the command, its identifier, and the custom payload itself.

Both the requests/responses and the list of commands the Digital Twin System is able to support can be represented with additional NGSI-LD Properties, as follows.

7.2.3.1 Property for listing available commands

The additional Property dedicated to the list of available commands is as follows:

```
"commands": {
  "type": "Property",
  "value": ["<cmd_name1>","<cmd_name2>", …, "<cmd_nameN>"]
}
```

It is a Property whose value is an array of Strings, each string representing the unique name of a supported command.

7.2.3.2 Properties for command endpoints

For each available command, a set of three endpoints is to be additionally created within the Digital Twin System, by means of three dedicated Properties per command. The first endpoint will manage that command's requests, the second endpoint will manage its status, and the third endpoint will manage command's results.

This convention dictates that:

- The NGSI-LD Property that manages requests has the same name as the command, e.g. "<cmd_name1>".
- The NGSI-LD Property that manages results has the same name as the command plus the "-RESULT" suffix.
- The NGSI-LD Property that manages status has the same name as the command plus the "-STATUS" suffix.

Each endpoint can receive multiple requests or responses, and it supports queueing of messages. For example, the command "moveToLocation" may receive a sequence of requests that are to be stored in an array and orderly processed depending on the arrival timestamps. A number of respective responses may be produced, as well. Thus, each endpoint, represented by its dedicated NGSI-LD Property, exploits the multi-Attribute feature (see ETSI GS CIM 009 [i.7], clause 4.5.5) as follows:

Command Request endpoint

```
"<cmd_name>": {
  "datasetId": a URI uniquely identifying the specific command request
               (optional, if the use case does not need command queueing),
  "type":      "Property",
  "qos":       an Integer, representing the desired QoS (optional, default=0),
  "value":     custom parameters of the command (mandatory)
}
```

Command Status endpoint

```
"<cmd_name>-STATUS": {
  "datasetId": a URI uniquely identifying the specific status feedback message
               (optional, if the use case does not need queueing them),
  "type":      "Property",
  "value":     custom status of the command (mandatory)
}
```

Command Result endpoint

```
"<cmd_name>-RESULT": {
  "datasetId": a URI uniquely identifying the specific result feedback message
               (optional, if the use case does not need queueing them),
  "type":      "Property",
  "value":     custom result of the command (mandatory)
}
```

Usually, the System Adapter (or the actuator behind it), upon receiving a command request with a specific "datasetId", will then generate status and result with the same "datasetId", so that, when the status/result is received by the application, it can link it back to the corresponding command that is generating the received feedback. The value of the request, status and result is generic, and it is up to the specific application to define useful values. As an example, the PackML language for the control of packaging machines [i.9] defines a set of possible values for statuses during an actuation workflow.

EXAMPLE 1:

An example follows, where the Digital Twin System represents a simple actuator. The example shows the NGSI-LD Entity representing a light that can change colour by manipulation of its brightness, hue and saturation values; further, it is possible to turn the lamp on and off. Apart from the **"id"** and the **"type"**, the actuator entity has a set of regular properties that represent the current status of the lamp. In the example these are "**colorRGB"** and "**is-on"**. Then it has the conventional Property named **"commands"**, signalling that it supports four commands: **"turn-on", "set-saturation", "set-brightness", "set-hue"**. Further, it has four (times three) additional properties serving the purpose of command endpoints.

```
{
  "id": "urn:ngsi-ld:pHueActuator:light1",
  "type": "Lamp",

  REGULAR PROPERTIES
  "colorRGB": {"type": "Property", "value": "0xABABAB"},
  "is-on": {"type": "Property", "value": true},

  AVAILABLE COMMANDS
  "commands": {
    "type": "Property",
    "value": ["turn-on", "set-saturation", "set-hue", "set-brightness"]
  }

  COMMAND ENDPOINTS
  "turn-on": {"type": "Property", "value": <custom request>}
  "turn-on-STATUS": {"type": "Property", "value": <custom status>}
  "turn-on-RESULT": {"type": "Property", "value": <custom response>}
  "set-hue": ...
  "set-hue-STATUS": ...
  "set-hue-RESULT": ...
  …
}
```

EXAMPLE 2:

The following example, shows an NGSI-LD Entity Fragment that can be used as a command request to request that the lamp be turned off.

```
{
  "id": "urn:ngsi-ld:pHueActuator:light1",
  "type": "Lamp",
  "turn-on": {
    "type": "Property",
    "qos": {
      "type": "Property",
      "value": 1
    },
    "value": false
  }
}
```

EXAMPLE 3:

In the following example, the value of the command request is a more complex JSON Object, to show that complex actions can be conveyed by just one request. Further, the request has an identifier that makes it possible to enqueue it, together with other request that may arrive to the same command endpoint within a timespan.

```
{
  "id": "urn:ngsi-ld:pHueActuator:light1",
  "type": "Lamp",
  "set-hue": {
    "type": "Property",
    "qos": {
      "type": "Property",
      "value": 1
    },
    "datasetId": "myapp:mycommand:1342",
    "value": {"red":"1 seconds", "green": "2 seconds"}
  }
}
```

In summary, the suggested convention prescribes a "**commands"** property that contains a list of commands supported by the actuator. For each of these commands, the convention requires a command endpoint consisting of three properties, the name of the command, e.g. "**turn-on**", the status property, which is the name of the command with "-**STATUS**" as suffix, and the result, which is the name of the command with "**-RESULT**" as suffix. Nevertheless, it is noted that such suffixes are just a convention to distinguish the endpoints. So far, two practical implementations exist, see clauses 7.2.5 and 7.2.6, that adopt the general scheme of this convention, with minor deviations. In fact, this convention is derived as a generalization that leverages the full potential of NGSI-LD sub-Attributes and multi-Attributes.

7.2.4 Communication model

7.2.4.1 Possible communication models

This convention can be leveraged by two different communication models:

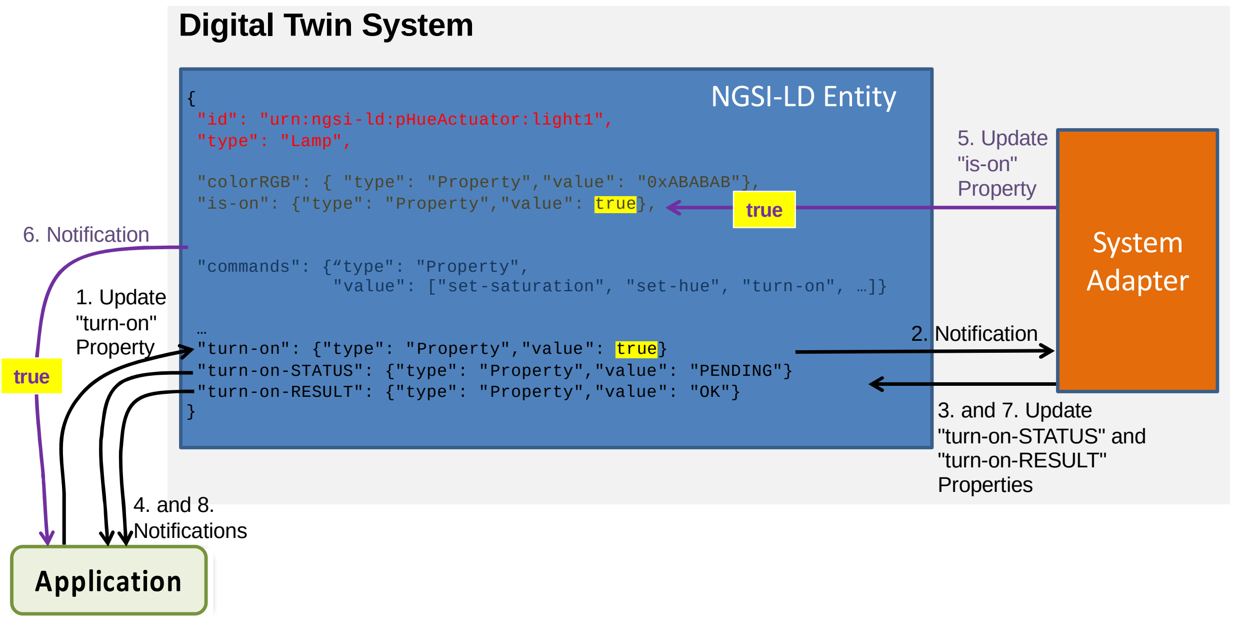
- Subscription/notification, where both the application and the System Adapter use NGSI-LD Subscriptions to have the command requests delivered to the appropriate handler within the System Adapter and vice-versa.
- Forwarding, which uses the NGSI-LD Registry and a System Adapter able to federate itself with the Context Broker holding the Digital Twin's Entity, as a means to deliver the commands.

7.2.4.2 Subscription/notification model

For the interaction to work, the System Adapter, acting as a proxy to the actuator, subscribes to all command properties; in EXAMPLE 1 of clause 7.2.3.2, these are "set-brightness", "set-saturation", "set-hue" and "turn-on". When the application, acting as the actuation client, updates the value of a command property, the System Adapter will receive the notification with the new value. This will be translated into the proprietary format and forwarded to the actuator using the actuator-specific protocol. The application in turn can subscribe to the command status and the result. The System Adapter updates the status of the actuation during the execution of the command, which is primarily relevant in the case of longer-lasting actuations, and finally updates the result, once the actuation has been completed. If the application has subscribed to the status and result, it will receive the corresponding notifications. Independent of the command-related properties, the status of the actuator, held within its regular properties, will be updated.

The detailed workflow is depicted in Figure 7.2.4.2-1, and can be interpreted as follows:

1) Application updates "turn-on" command Property with "value": true
2) System Adapter gets notification of the new value true
3) System Adapter updates "turn-on-STATUS" command Property with "value": "PENDING"
4) Application gets notification of the new value "PENDING"
5) System Adapter updates "is-on" regular Property with value: true
6) Application gets notification with value: true
7) System Adapter updates "turn-on-RESULT" command Property with "value": "OK"
8) Application gets notification with of the new value "OK"


Digital Twin System
NGSI-LD Entity
5. Update "is-on" Property
true
System Adapter
6. Notification
1. Update "turn-on" Property
true
2. Notification
3. and 7. Update "turn-on-STATUS" and "turn-on-RESULT" Properties
4. and 8. Notifications
Application

```
{
 "id": "urn:ngsi-ld:pHueActuator:light1",
 "type": "Lamp",

 "colorRGB": { "type": "Property","value": "0xABABAB"},
 "is-on": {"type": "Property","value": true},


 "commands": {“type": "Property",
             "value": ["set-saturation", "set-hue", "turn-on", …]}

 …
"turn-on": {"type": "Property","value": true}
"turn-on-STATUS": {"type": "Property","value": "PENDING"}
"turn-on-RESULT": {"type": "Property","value": "OK"}
}
```

Figure 7.2.4.2-1: Steps of the actuation workflow using subscription/notification

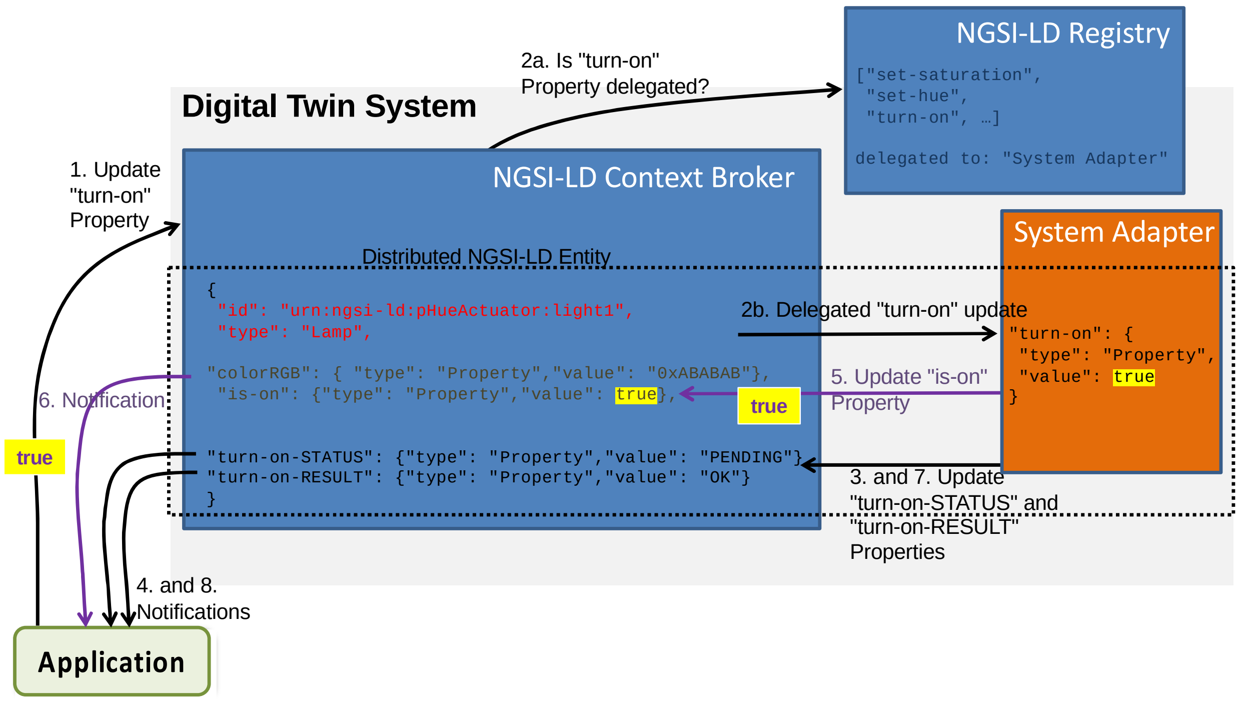
7.2.4.3 Forwarding model

The forwarding model uses registrations and forwarding of requests. Actuation of commands is provisioned via registration(s) to the NGSI-LD Registry done by the System Adapter that states "I am responsible for command property <X>". When the Application changes the value of a command property, first the NGSI-LD Context Broker asks to the NGSI-LD Registry whether the property is delegated to some other component. The NGSI-LD Registry knows that property <X> of the Entity is delegated to the System Adapter. Hence, the request is forwarded to the System Adapter. Similar to the other communication model, the request will then be translated into the proprietary format and forwarded to the actuator using the actuator-specific protocol.

In this model, the NGSI-LD Entity is distributed over two different components, because some of its properties live in the Context Brokers and other properties live in the System Adapter, as indicated in Figure 7.2.4.3-1 with a dotted rectangle.

The rest of the workflow, i.e. delivery of status and result messages to the application, is done similarly to the subscription/notification model. The detailed workflow is depicted in Figure 7.2.4.3-1, and can be interpreted as follows:

1) Application updates "turn-on" command Property with "value": true
2) Context Broker ask Registry where to forward the request
3) Context Broker forwards request to System Adapter
4) System Adapter updates "turn-on-STATUS" command Property with "value": "PENDING"
5) Application gets notification of the new value "PENDING"
6) System Adapter updates "is-on" regular Property with value: true
7) Application gets notification with value: true
8) System Adapter updates "turn-on-RESULT" command Property with "value": "OK"
9) Application gets notification with of the new value "OK"


Digital Twin System
NGSI-LD Registry
["set-saturation",
"set-hue",
"turn-on", …]
delegated to: "System Adapter"
2a. Is "turn-on" Property delegated?
NGSI-LD Context Broker
1. Update "turn-on" Property
Distributed NGSI-LD Entity
System Adapter
2b. Delegated "turn-on" update
"turn-on": { "type": "Property", "value": true }
5. Update "is-on" Property
6. Notification
true
true
3. and 7. Update "turn-on-STATUS" and "turn-on-RESULT" Properties
4. and 8. Notifications
Application

```
{
  "id": "urn:ngsi-ld:pHueActuator:light1",
  "type": "Lamp",

 "colorRGB": { "type": "Property","value": "0xABABAB"},
  "is-on": {"type": "Property","value": true},

 "turn-on-STATUS": {"type": "Property","value": "PENDING"}
 "turn-on-RESULT": {"type": "Property","value": "OK"}
}
```

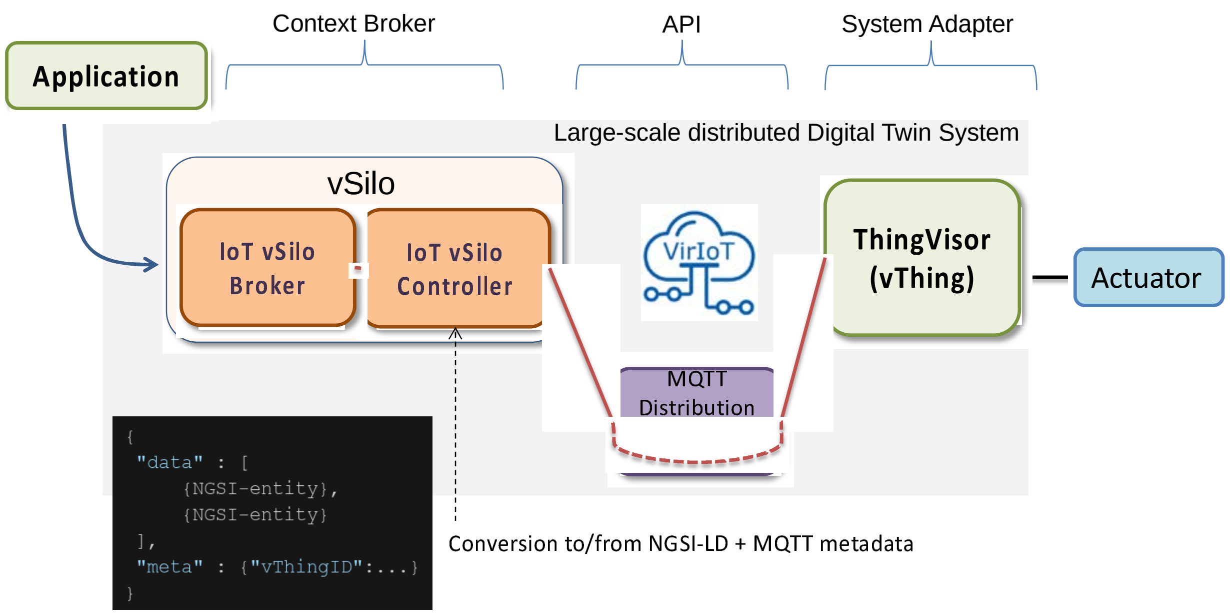
Figure 7.2.4.3-1: Steps of the actuation workflow using forwarding

7.2.5 Implementation of the subscription-based actuation workflow

The Fed4IoT project (https://fed4iot.org) leverages the NGSI-LD architecture and the subscription/notification workflow for actuation, in order to implement the concept of a Cloud of Things. It enables virtualization of existing IoT sensors/actuators through Virtual Things and IoT Brokers. IoT application developers can simply rent the Virtual Things and the Brokers their applications need.

The Fed4IoT's Cloud of Things is named VirIoT (https://github.com/fed4iot/VirIoT), and it is based on the concept of Virtual Silos as-a-service: isolated and secure IoT environments made of Virtual Things whose data can be accessed through standard IoT Brokers.

In Figure 7.2.5-1 a diagram shows how VirIoT implements the concept of a large-scale and distribute Digital Twin System that leverages the architecture and the workflow convention described in clause 7.2.4.2.



```
{
 "data" : [
      {NGSI-entity},
      {NGSI-entity}
 ],
 "meta" : {"vThingID":...}
}
```

Figure 7.2.5-1: VirIoT implementation of the Digital Twin System and actuation workflow

All components encapsulate requests in a neutral-format message that leverages NGSI-LD Entities at its core. But, since VirIoT uses MQTT as its internal data distribution system, all information and actuation commands are encoded as NGSI-LD entities, plus an additional "meta header" that is used by the MQTT to publish and subscribe in a broadcast fashion to multiple vThings, because the same virtual sensor can be used by multiple applications at the same time.

For the actuation workflow, the "data" part of this message contains the command request, as specified in clause 7.2.3, but with an additional value key that is the "command notification uri" (`cmd-nuri`), representing a location where feedback (status, result) should be sent by the ThingVisor. For example, the `cmd-nuri` contains the "data_in" MQTT topic of the issuing vSilo, so that command feedback (status and results) is sent to it, only, instead of being broadcasted to all subscribing applications.

VirIoT is agnostic to access control issues to a virtual actuator, since the relevant policies are implemented in the specific ThingVisor, which can grant tokens to execute actuation-commands to a subset of vSilos only, through preliminary exchange of specific actuation-commands (a kind of log-in).

Fed4IoT has developed several different ThingVisors (System Adapters for different sensors and hardware): for example, lamp systems and robot devices are virtualized through specific ThingVisors, and applications can control the lighting system of a rented conference room or control camera and position of a bot by adding related virtual actuators to their vSilo.

7.2.6 Implementation of the registration-based actuation workflow

The IoT Agent node library [i.1] introduces the concept of an IoT Agent, which is a component that lets a group of devices send their data to and be managed from a Context Broker using their own native protocols. Thus, it corresponds to the System Adapter, and wires up the IoT devices so that measurements can be read and commands can be sent using NGSI-LD requests sent to an NGSI-LD compliant context broker.

IoT Agents already exist or are in development for many IoT communication protocols and data models. Examples include the following:

- IoTAgent-JSON - a bridge between HTTP/MQTT messaging (with a JSON payload) and NGSI-LD
- IoTAgent-LWM2M - a bridge between the Lightweight M2M protocol and NGSI-LD
- IoTAgent-UL - a bridge between HTTP/MQTT messaging (with an UltraLight2.0 payload) and NGSI-LD
- IoTagent-LoRaWAN - a bridge between the LoRaWAN protocol and NGSI-LD

This implementation follows the communication model described in clause 7.2.4.3, as explained in Figure 7.2.6-1. In this workflow:

- Requests between User and Context Broker use NGSI-LD
- Requests between Context Broker and IoT Agent use NGSI-LD
- Requests between IoT Agent and IoT Device use native protocols
- Requests between IoT Device and IoT Agent use native protocols
- Requests between IoT Agent and Context Broker use NGSI-LD

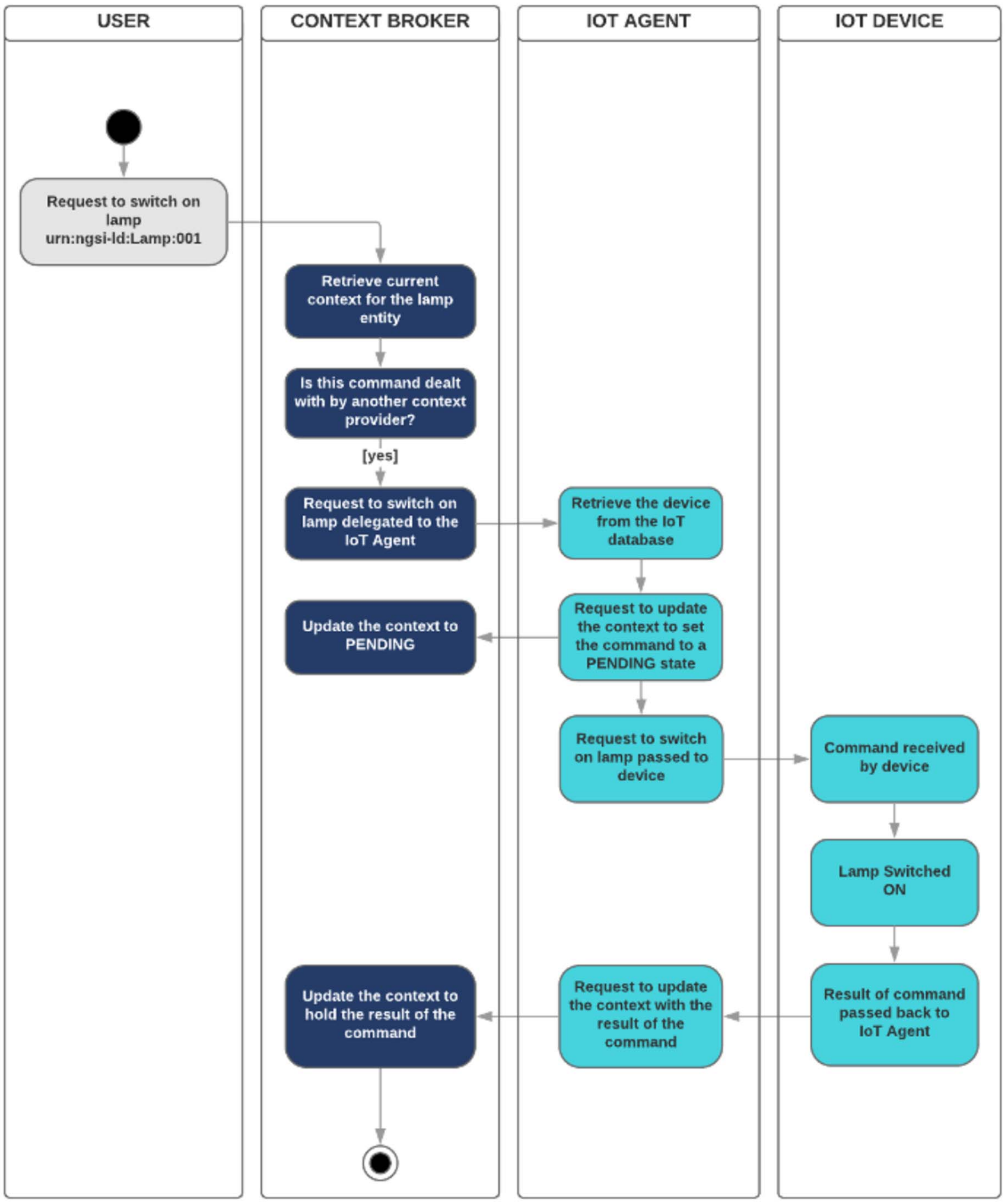

USER
CONTEXT BROKER
IOT AGENT
IOT DEVICE
Request to switch on lamp urn:ngsi-ld:Lamp:001
Retrieve current context for the lamp entity
Is this command dealt with by another context provider?
[yes]
Request to switch on lamp delegated to the IoT Agent
Retrieve the device from the IoT database
Update the context to PENDING
Request to update the context to set the command to a PENDING state
Request to switch on lamp passed to device
Command received by device
Lamp Switched ON
Update the context to hold the result of the command
Request to update the context with the result of the command
Result of command passed back to IoT Agent

Figure 7.2.6-1: IoT Agent node library implementation of the Digital Twin System and actuation workflow

Provisioning of the devices will be carried out (via REST API) through IoT Agents, as well. This provisioning implies that, on the one hand, the corresponding entities (with their commands), that represent the devices, are generated in the Context Broker and, on the other hand, that the corresponding IoT Agent is configured for communication with the corresponding device, all in one provisioning step. Below, an example how to provision a device which supports start and stop commands is presented.

```
{
    "devices": [
        {
            "device_id":   "device001",
            "entity_name": "urn:ngsi-ld:Device:001",
            "entity_type": "Device",
            "attributes": [
                { "object_id": "s", "name": "isOpen", "type": "boolean" }
            ],
            "commands": [
                { "name": "start", "type": "command" },
                { "name": "stop", "type": "command" }
            ],
            "static_attributes": [
                {"name":"name", "type": "Text","value": "Device:001 provision"}
            ]
        }
    ]
}
```

7.3 Service execution with Digital Twins

7.3.1 Motivation

To realize the Digital Twin functionalities as described in clause 6.2 and integrate them with the NGSI-LD functionality, e.g. as part of a Context Broker, as opposed to implementing everything as part of an external application logic, a more explicit representation is required. In the approach described in clause 7.2, each component independently updates and subscribes to commands, status and results. If multiple applications are involved, the synchronization between these components has to be implemented by the components themselves. This is especially problematic if more applications are added at a later time. In such a scenario, it is difficult to enforce priorities, e.g. in an autonomous driving example, a component having predicted a safety critical situation should have higher priority than one trying to optimize fuel consumption. Also, there may be dependencies, e.g. actions should only be taken if preconditions are met or previous actions having been completed. To do this in a reliable way, more support by a system component is required.

7.3.2 Solution proposal

The proposal is to make service execution an explicit functionality of NGSI-LD - this functionality can then be used to trigger diagnosis, prediction, simulation and in particular actuation. Entities in addition to properties and relationships would have services. Applications would explicitly create service executions, including means for defining complex service executions like sequences or conditional service executions, and also event-triggered service executions could be provided, which, especially in the latter case, could significantly reduce delay as it is not required to first go back to an external application.

Figure 7.3.2-1 shows two additional architectural roles for NGSI-LD, the Service Execution Application and the Service Executor. The Service Executor has a similar role as the Context Source, but instead of providing context information, it enables the execution of services related to one or more entities/Digital Twins. The Service Execution Application is an application that creates service executions. As these are architectural roles, real applications can of course have both roles, Context Consumer and Service Executing Application, at the same time.

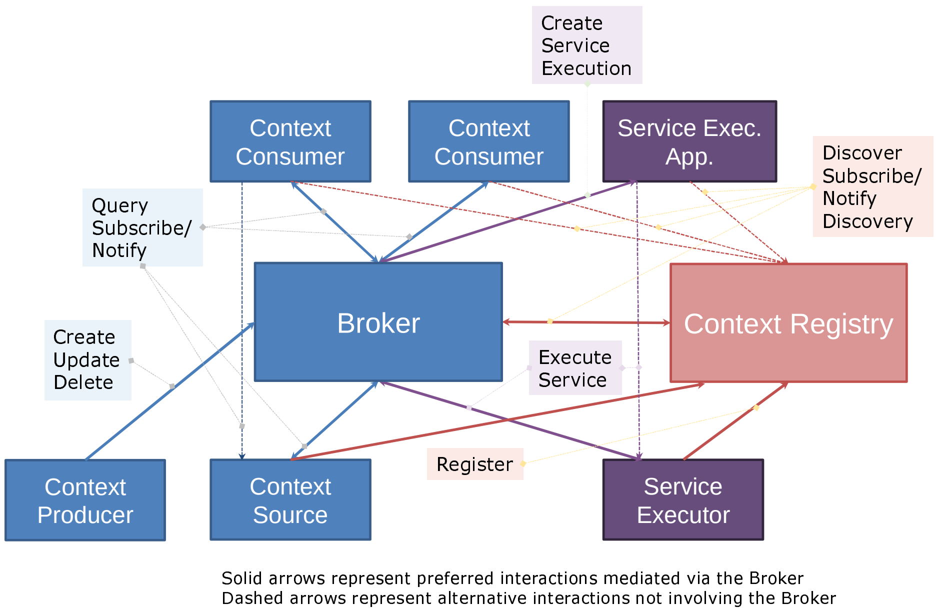


Figure 7.3.2-1: Additional architectural roles for NGSI-LD (Service Execution Application and Service Executor)

7.3.3 Service Executor

- A Service Executor has to provide an http POST interface, which takes a JSON payload providing the parameters needed for service execution.
- A Service Executor registers itself with the Registry Server (in analogy to a Context Source).
- A Service Executor registration includes the following aspects:
  - Service endpoint (URL)
  - Entity (Id)
  - Services it can execute with respective parameters
  - Additional properties, e.g. geographic location, owner, …

7.3.4 Service execution

A new endpoint for entities services is proposed on `/entities/{entityId}/services/{serviceName}`. The interactions required for executing a service are shown in Figure 7.3.4-1:

- Service Executing Application creates Service Execution instance, providing the relevant parameters and dependencies
- Broker queries Service Executor from Registry Server
- Broker calls Service Executor providing the relevant parameters
- Service Execution instance is updated with status in case of asynchronous execution

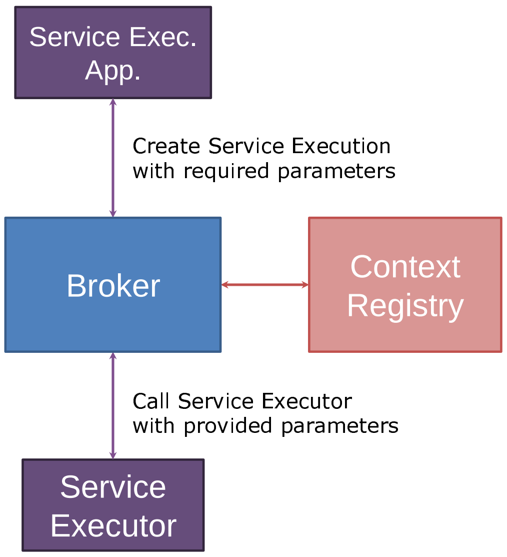

Service Exec. App.
Create Service Execution with required parameters
Broker
Context Registry
Call Service Executor with provided parameters
Service Executor

Figure 7.3.4-1: Functionality for sequential/parallel execution of services, conditional execution and event-triggered service execution

7.3.5 Service execution example

In this clause, a simple example for the service execution is provided. The syntax is to be considered preliminary, as more advanced aspects such as ordered execution, parallel execution, conditional ordered execution, conditional execution triggered by notification and the use of property values or relationship objects as input parameters still needs to be integrated.

In the first step, a Service Executor registers itself with a Context Registry (enhanced to be able to deal with Service Executor registrations).

```
{
    "id": "urn:ngsi-ld:serviceRegistration:csr800949",
    "type": "ServiceRegistration",
    "name": "ServiceRegA",
    "description": "DescriptionExample",
    "information": [
        {
            "services": [
                {
                    "id": "urn:ngsi-ld:room:A0815",
                    "name": "lightswitch",
                    "input": {
                        "type": "attribs",
                        "attribs": [
                            {
                                "attribname": "state",
                                "datatype": "boolean",
                                "range": [
                                    true,
                                    false
                                ]
                            }
                        ]
                    },
                    "output": {
                        "type": "attribs",
                        "attribs": [
                            {
                                "attribname": "success",
                                "datatype": "boolean",
                                "value": [
                                    true,
                                    false
                                ]
                            }
                        ]
                    },
                "endpoint": "http://localhost:/callmehere/lightswitch"

    ],
    "location": {"type": "Polygon", "coordinates":
[[[8.686752319335938,49.359122687528746],[8.742027282714844,49.3642654834877],[8.767433166503904,49.
398462568451485],[8.768119812011719,49.42750021620163],[8.74305725097656,49.44781634951542],[8.66924
2858886719,49.43754770762113],[8.63525390625,49.41968407776289],[8.637657165527344,49.3995797187007]
,[8.663749694824219,49.36851347448498],[8.686752319335938,49.359122687528746]]] }
}
```

As a result of this registration, the entity identified as `urn:ngsi-ld:room:A0815` has a service with the name `lightswitch`, as shown in the example below.

```
{
"id": "urn:ngsi-ld:room:A0815"
    "type": "Room"
    …
    "services": [
{
"name": "lightswitch",
            "input": {
                "type": "attribs",
                "attribs": [
                    {
                        "attribname": "state",
                        "datatype": "boolean",
                        "range": [
                            true,
                            false
                        ]
                    }
                ]
            },
            "output": {
                "type": "attribs",
                "attribs": [
                    {
                        "attribname": "success",
                        "datatype": "boolean",
                        "value": [
                            true,
                            false
                        ]
                    }
                ]
            }
        ]
}
```

The Service Execution Application can now create a service request that can contain a set of executions, in this case just a single one for switching on the light, using the `lightswitch` service registered preciously for entity `urn:ngsi-ld:room:A0815`.

```
{
    "type": "ServiceRequest",
    "id": "urn:servicerequest1",
    "executions": [{
        "id": "urn:execrequest1",
        "name": "lightswitch",
        "entityId": "urn:ngsi-ld:room:A0815",
        "entityType": "Room",
        "type": "ExecutionRequest",
        "input": {
            "type": "attribs",
            "attribs": [{
                "attribname": "state",
                "datatype": "boolean",
                "value": true
            }]
        }
    }]
}
```

The service requests triggers the execution of switching on the light by setting `state` to `true`, which is being forwarded to the Service Executor. This may result in the following response, which is used to update the service request `urn:servicerequest1`. Intermediate status updates can be provided there as well.

```
    {
        "name": "lightswitch",
        "entityId":"urn:ngsi-ld:room:A0815",
        "success": {
                    "datatype": "boolean",
                    "value": true
        },
        "type": "ServiceResult"
    }
```

The status of a service executions related to a service request can be requested. An example with multiple executions is shown below:

```
{
    "id": "{serviceId}",
    "executions": [{
            "id": "exec1",
            "status": "failed"
        },
        {
            "id": "exec2",
            "status": "success"
        },
        {
            "id": "exec3",
            "status": "running"
        },
        {
            "id": "exec4",
            "status": "pending"
        }
    ]
}
```

Annex A:
Change History

Date	Version	Information about changes
2020-10	0.0.1	Early draft (publicly shared at https://docbox.etsi.org/ISG/CIM/Open/drafts/)
2021-06	0.0.2	New draft (publicly shared at https://docbox.etsi.org/ISG/CIM/Open/drafts/)
2022-07	0.0.3	Advanced draft (publicly shared at https://docbox.etsi.org/ISG/CIM/Open/drafts/)
2022-10	0.0.4	Final draft
2022-11	1.1.1	ISG CIM approval and Technical Officer review for pre-publication processing

History

Document history		
V1.1.1	December 2022	Publication