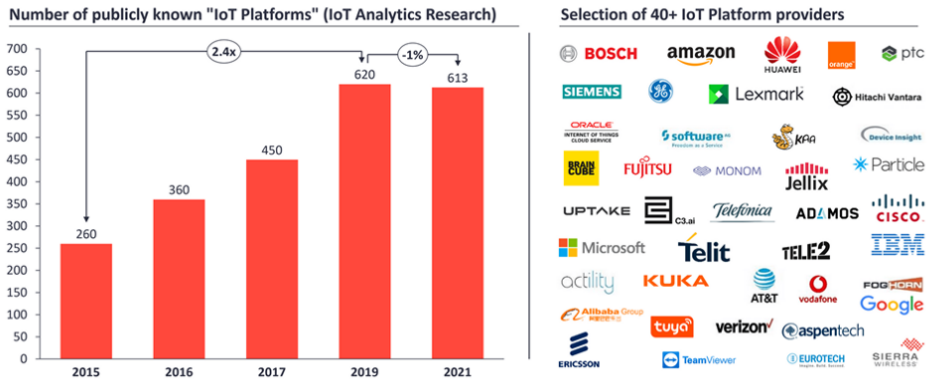
5G Personal IoT Network(s) (PINs) and oneM2M IoT Service Layer (SL) Platform

IOT ANALYTICS

Your Global IoT Market Research Partner

Number of publicly known "IoT Platforms" (2015-2021)

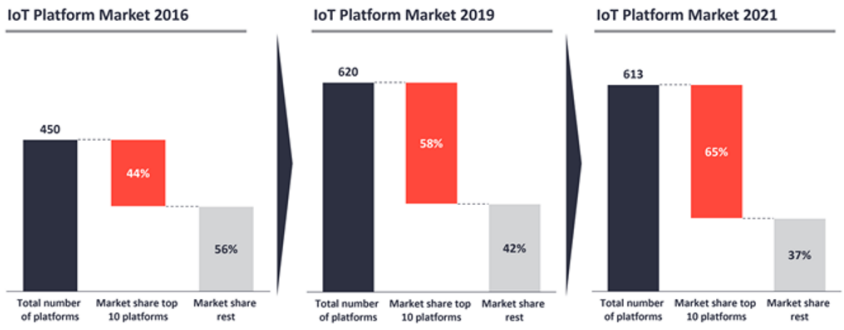


Source: IoT Analytics Research 2021; Note: IoT Analytics' definition of an IoT Platform has shifted slightly over time. Condition for republishing: Source citation with link to original post and company website; Non-commercial purposes only

IOT ANALYTICS

Your Global IoT Market Research Partner

Concentration in the IoT Platform Market 2016 to 2021



Source: IoT Analytics Research 2021; Condition for republishing: Source citation with link to original post and company website; Non-commercial purposes only

Five (5) Sensor Technologies are set to change the IoT Sensor landscape in the coming years, i.e., 1. Smarter Sensors; 2. more Power-efficient Sensors; 3. Soft & Virtual Sensors; 4. Sensor Fusion; 5. Biosensors. The average IoT Device now comes with four (4) sensors. The increasing penetration of IoT Devices is leading to different Technical Requirements for Sensor Manufacturers (e.g., Ultra-Low Power, Smaller Form Factor, Pre-processing, & Connectivity.

"Smart-Sensors" (embedded ML inference capability) Proliferation

Key IoT sensor technology innovations include a much higher computing capacity and the ability to detect signals from multiple discrete sensing elements. The industry refers to these more advanced devices as "smart sensors." Instead of simply passing on the sensor signals to the next level in the value chain, smart sensors can process signals directly (e.g., validating and interpreting the data, displaying the results, or running specific analytics applications); in this way, sensors become edge devices. **The "most advanced Smart Sensors" are now also incorporating AI into their design**. These Sensors are designed for AI inference, which has numerous advantages, e.g., decisions can be made immediately, & sensitive Data can be processed without sending it elsewhere & creating the risk of Data theft.

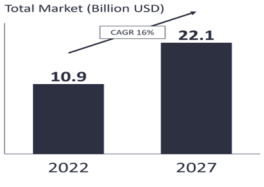
IOT ANALYTICS

January 2023

Your Global IoT Market Research Partner

Market Snapshot: IoT Sensor Market 2022

Market Size



Leading vendors (selection)

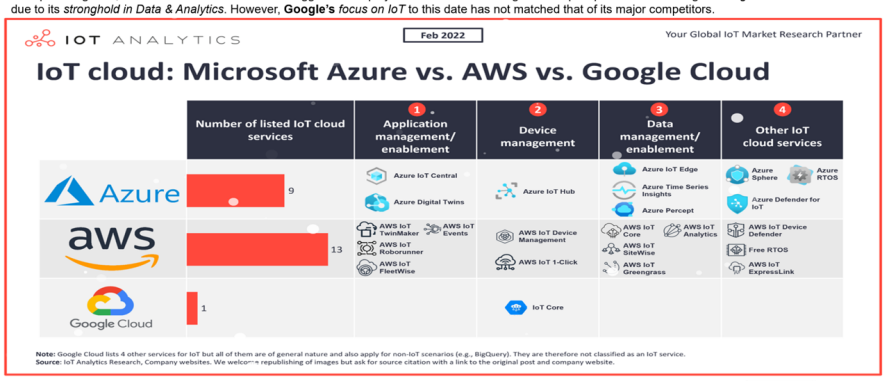
SONY, BOSCH, ams, ST, TELEDYNE TECHNOLOGIES, OMNIVISION, Infineon + CYPRESS, SENSIRION, Sensata, ON, Honeywell, TE, muRata, knowles, NXP, TDK, ANALOG DEVICES, Amphenol, BROADCOM, Texas Instruments, RENESAS, AKM, SMARTSENS

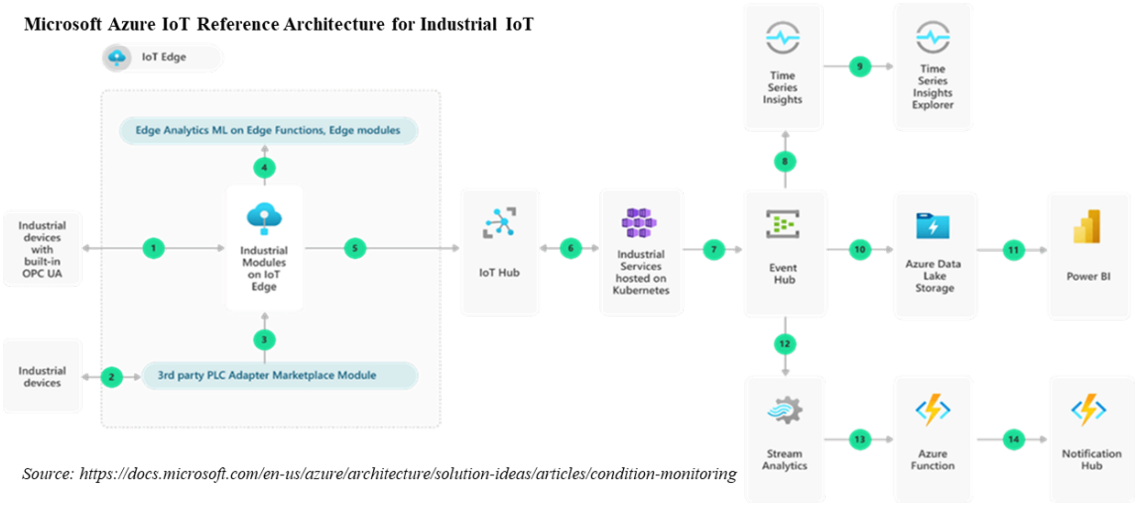
5 trending technologies

1	Smart sensors
2	Power-efficient sensors
3	Soft & Virtual sensors
4	Sensor fusion
5	Biosensors

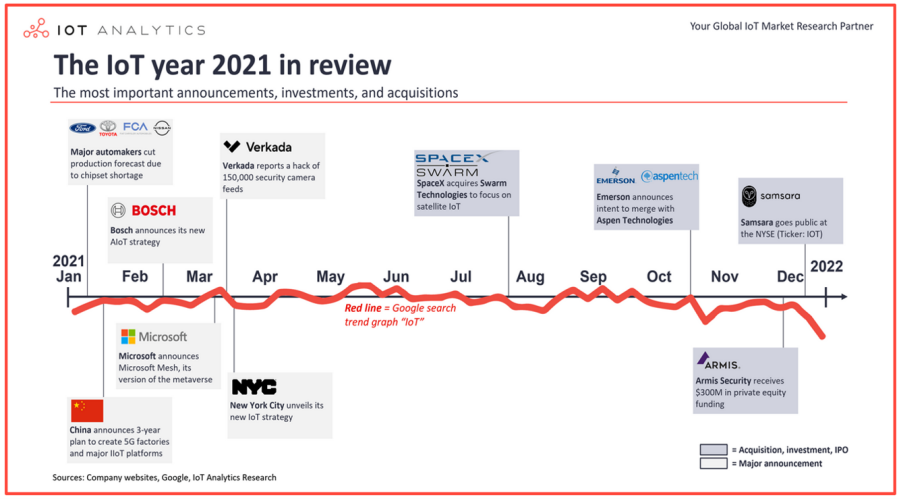
Source: IoT Analytics Research, IoT Sensor Market Report 2022-2027. We welcome republishing of images but ask for source citation with a link to the original post and company website.

The three leading Global Hyperscalers (**AWS**, **Microsoft**, & **Google Cloud**) **hold more than 80% Market share for Global Public Cloud Services specifically for IoT Workloads** according to the latest research from IoT Analytics: Cloud Computing for IoT Market Report 2021–2026… Although offering Public Cloud Services for connecting IoT Devices is not (yet) a major driver for the Businesses of these Companies, IoT Services & related Cloud Infrastructure Investments are expected to grow faster than general Public Cloud spending and will thus increase in strategic importance during the coming years as Public Cloud becomes a multi-trillion-dollar market. **AWS** *had the 1st-mover advantage, introducing Public Cloud Services in 2006* and *adding IoT-specific Services in 2015.* **Microsoft Azure**, which *launched its Public Cloud four (4) years after AWS, introduced its IoT services only five months after AWS* and has since managed to overtake **AWS** *as the market leader for IoT-specific Services thanks to its IoT-centric strategy, a $5 billion investment in IoT in 2018*, and its strong business footprint in the Enterprise segment, which accounts for some of the biggest IoT deployments. **GCP** has managed to keep its position as a strong *challenger to* ***Azure*** *and* ***AWS*** due to its *stronghold in Data & Analytics*. However, **Google's** *focus on IoT* to this date has not matched that of its major competitors.





Source: https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/condition-monitoring



oneM2M Overview

The oneM2M Global Organization creates Technical Specifications (TSs) to ensure that Machine-to-Machine (M2M) Communications can effectively operate on a Worldwide scale.

Seven (7) of the World's leading Information and Communications Technology (ICT) Standards Development Organizations (SDOs) launched in July 2012 a new Global Organization to ensure the most efficient Deployment of Machine-to-Machine (M2M) Communications Systems.

The new organization, called **oneM2M**, develops specifications to ensure the Global Functionality of M2M—allowing a range of Industries to effectively take advantage of the benefits of this emerging Technology.

The seven (7) majors ICT SDO founders of oneM2M are:

- The European Telecommunications Standards Institute (ETSI) , Europe
- The Association of Radio Industries and Businesses (ARIB), Japan
- The Telecommunication Technology Committee (TTC), Japan
- The Alliance for Telecommunications Industry Solutions (ATIS), USA
- The Telecommunications Industry Association (TIA), USA
- The China Communications Standards Association (CCSA), China
- The Telecommunications Technology Association (TTA), Korea

The members of the organization are devoted to developing Technical Specifications and Reports to ensure M2M Devices can successfully communicate on a Global scale.

The oneM2M Standardization work is split in five (5) WG:

- WG1: Requirements
- WG2: Architecture
- WG3: Protocols
- WG4: Security
- WG5: Requirements and Domain Models (former MAS-Management, Abstraction and Semantics)

The Test Specifications cover different testing aspects such as interoperability, interworking, Conformance, Performance and Security for different Protocols and Network elements.

Akraino IoT oneM2M Contact Person

Akraino oneM2M Contact person - Ike Alisson

oneM2M supported IoT Use Cases (UCs) - SAREF (**S**mart **A**pplications **REF**erence) Ontology

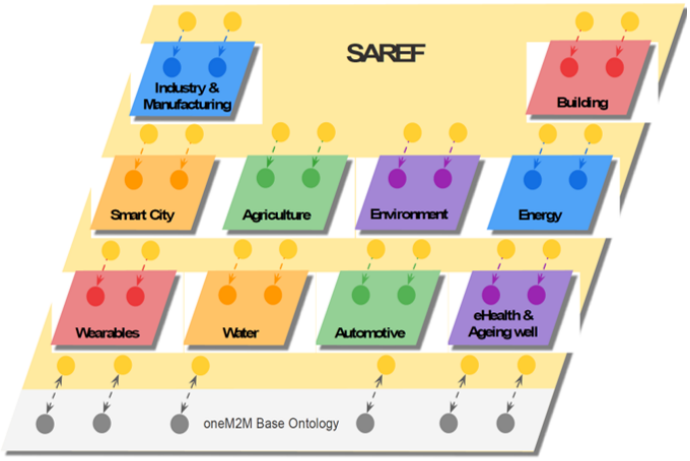
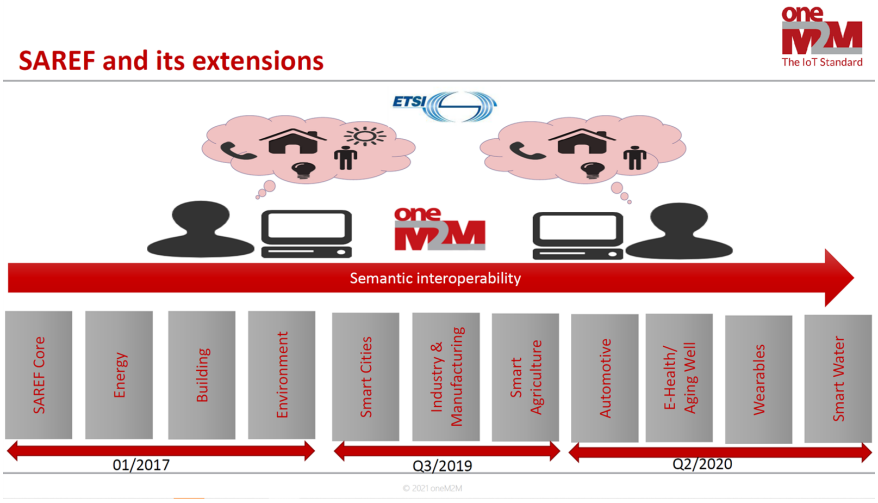


Figure 11: SAREF and its extensions



oneM2M supports IoT Use Cases (UCs) Smart M2M **SAREF** (**S**mart **A**ppliances **REF**erence) Ontology. Initially, SAREF V1 was common to three (3) Domains, namely:

1. Energy,
2. Environment
3. Buildings

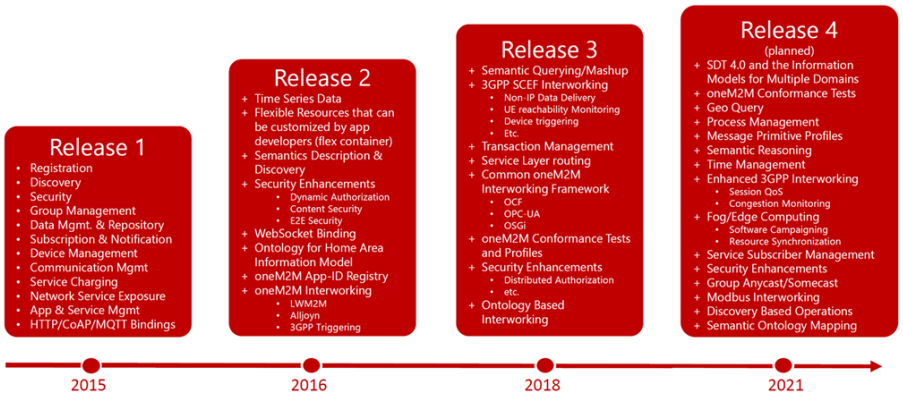
The first core of SAREF (mapped into three (3) Applications' Domains) has been improved to SAREF V2, thento SAREF V3 to enable mapping of SAREF with more Smart Applications domains such as:

4. Smart City,
5. Smart Industry and Manufacturing
6. Smart Agriculture and Food,
7. Automotive
8. eHealth and Ageing-Well,
9. Wearables
10. Smart Water,
11. Smart Lift…)

Like this **SAREF** became **S**mart **A**pplications **REF**erence Ontology (core **SAREF**) with its domain mapping extensions.

oneM2M Release Roadmap

oneM2M Feature Summary by Release



oneM2M Future Feature development

- oneM2M Release 5
  - Use Case and Requirements development
    - Work ongoing in Requirements and Domain Models Working Group
  - Architecture and protocol related work – to be started Q1 2021

Release 5 Work in Progress
oneM2M System Enhancements to Support Data Protection Regulations [WI-0095]
Effective IoT Communication to Protect 3GPP Networks [WI-0096]
oneM2M and SensorThings API [WI-0100]
Advanced Semantic Discovery [WI-0101]
System enhancements to support Data License Management [WI-0102]
...

oneM2M Layered Architecure Model

oneM2M Layered Model comprises three (3) layers:

- the Application Layer,
- the Common Services Layer
- the underlying Network Services Layer.

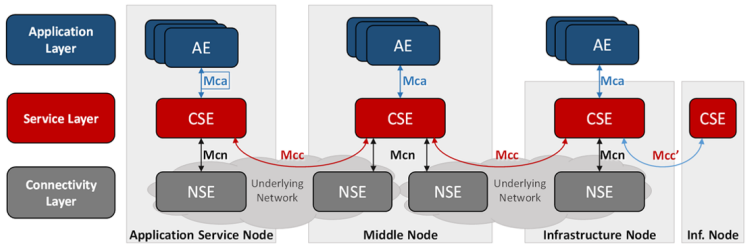


Figure 5.1.2-1: oneM2M Layered Model

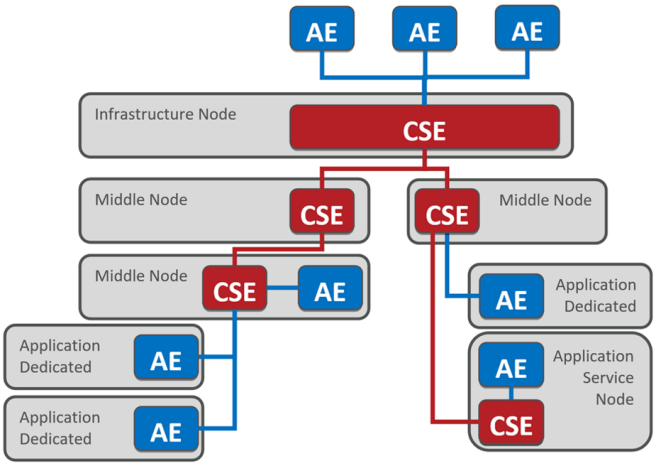


Figure 5.1.2-3: oneM2M node topology

Application Entity (AE): The Application Entity is an Entity in the Application Layer that implements an M2M Application Service Logic. Each Application Service Logic can be resident in a number of M2M Nodes and/or more than once on a Single M2M Node. Each execution instance of an Application Service Logic is termed an "Application Entity" (AE) and is identified with a unique AE-ID.

Examples of the AEs include an instance of a fleet tracking application, a remote blood sugar measuring application, a power metering application or a pump controlling application.

Common Services Entity (CSE): A Common Services Entity represents an Instantiation of a Set of "Common Service Functions" of the oneM2M Service Layer. A CSE is actually the Entity that contains the collection of oneM2M-specified Common Service Functions that AEs are able to use. Such Service Functions are exposed to other Entities through the **Mca** (exposure to AEs) and **Mcc** (exposure to other CSEs) Reference Points.

Reference Point **Mcn** is used for accessing services provided by the underlying Network Service Entities (**NSE**) such as waking up a sleeping device. Each **CSE** is identified with a unique CSE-ID.

Examples of service functions offered by the CSE include: data storage & sharing with access control and authorization, event detection and notification, group communication, scheduling of data exchanges, device management, and location services.

Network Services Entity (NSE): A Network Services Entity provides Services from the underlying Network to the CSEs.

oneM2M Common Service functions (CSFs) (applied to all IoT Domains: SAREF IoT UCs (Use Cases))

oneM2M's horizontal Architecture provides a Common Framework for IoT. oneM2M has identified a Set of Common Functionnalities, that are **applicable to all the IoT Domains (SAREF)**. Think of these functions as a large Toolbox with special tools to solve a number of IoT problems across many different domains. Very much like a screw driver can be used to fasten screws in a car as well as in a plane, the oneM2M **CSFs** are applicable to different IoT Use Cases (UCs)/Domains in different Industry Domains.

In its first phase, oneM2M went through a large number of IoT Use Cases (UCs) and identified a Set of Common Requirements which resulted in the Design of this Set of Tools termed **Common Service Functions (CSFs)**.

Furthermore, oneM2M has standardized how these Functions are being executed, i.e. is has defined uniform APIs to access these functions. Figure 5.3.1-1 shows a grouping of these functions into a few different scopes.

Figure 5.1.3-1: Common Service Functions

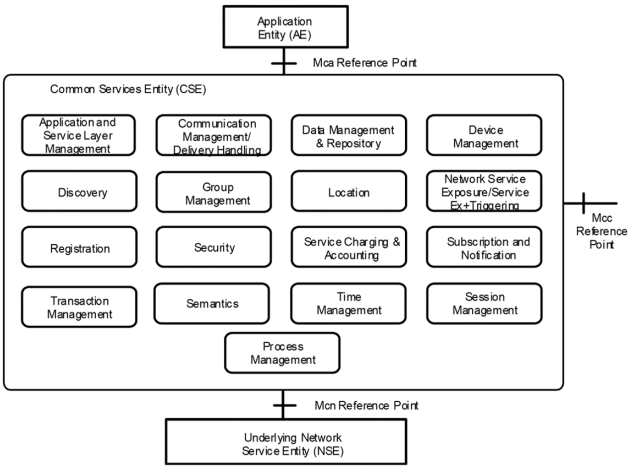


Figure 6.2.0-1: Common Services Functions

The Services above reside within a CSE (Common Service Entity) and are referred to as **Common Services Functions (CSFs).** The CSFs provide Services to the AEs via the Mca Reference Point and to other CSEs via the Mcc Reference Point.

oneM2M pre-integrated with 5G (3GPP) Specifications for cIoT

oneM2M baseline Architecture supports interworking with 3GPP and 3GPP Cellular Internet of Things (CIoT) Features such as IP and non-IP Data Control Plane Data Delivery. The oneM2M system may leverage the IoT related Features and Services that 3GPP added in Releases 10 through 15. Features and Services may be accessed by an ADN-AE, MN-CSE, or an ASN-CSE that is hosted on a UE and an IN-CSE that is able to access services that are exposed by a MNO.

The “3GPP Trust Domain” in Figure 5.2-1 captures the Functional entities that shall be part of the 3GPP Domain (the Network). Although the Figure 5.2-1 shows that the IN-CSE and IN-AE are outside of the 3GPP Domain, the IN-CSE may be part of the Operator domain (Fig. 4.2-1b).

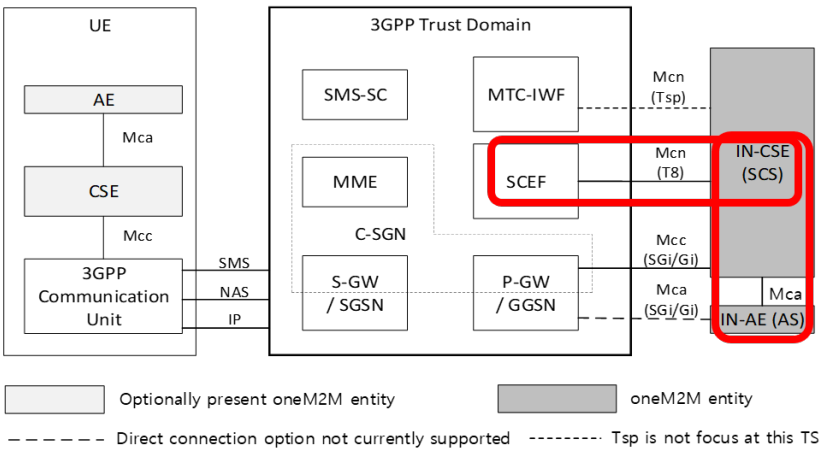


Figure 5.2-1: oneM2M Interfaces to the underlying 3GPP Network

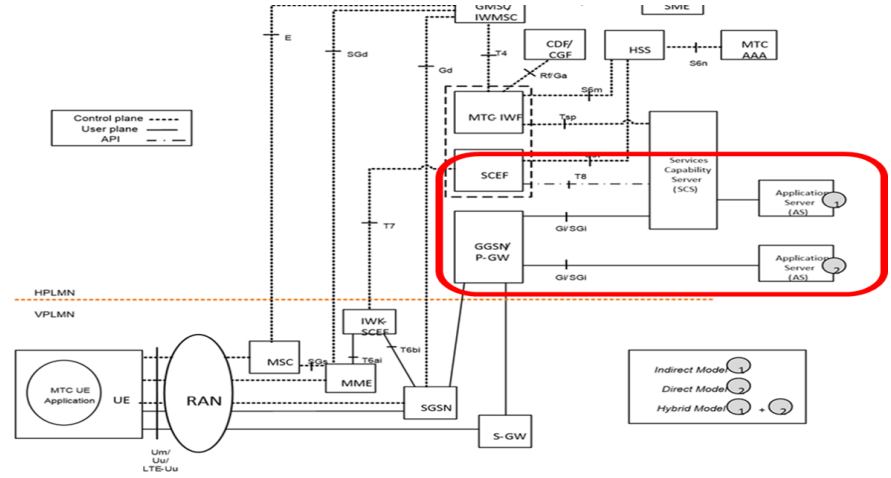


Figure 4.2-1b: 3GPP Architecture for Machine-Type Communication (Roaming)

The 3GPP Trust Domain provides three (3) Interfaces to SCS/CSE (Service Capability Server/Common Service Entity) for MTC:

i) IP based interface at SGi reference point,

ii) RESTful API interface at T8 Reference Point,

iii) Diameter based interface at Tsp Reference Point.

The Service Capability Server (SCS) is a 3GPP term that refers to an entity which connects to the 3GPP Trust Domain to communicate with UEs used for Machine Type Communication (MTC).

oneM2M Cloud Vendor Independent

oneM2M is **Cloud Provider independent:** From Fragmentation to Standards and decoupling Device, Cloud, and Application by Open Interfaces.

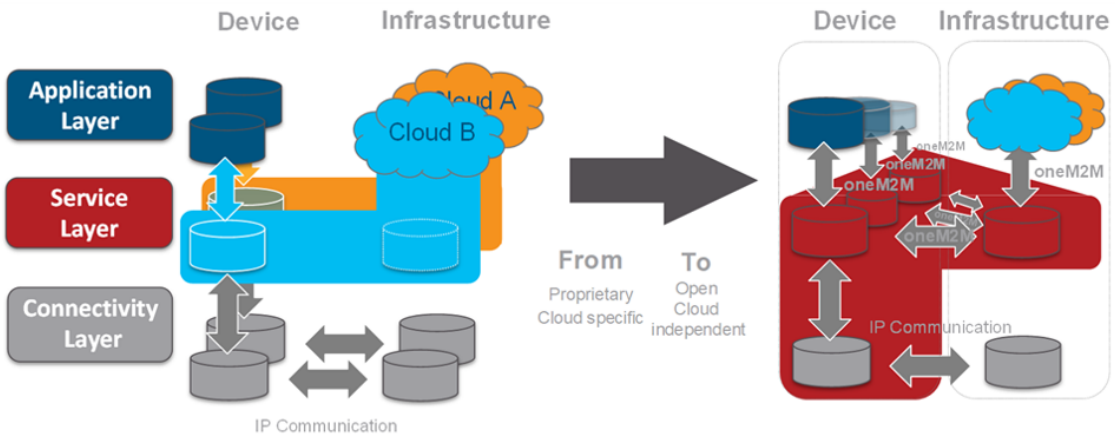


Figure 5.1.1-3: Cloud provider independent

oneM2M addresses Edge/Fog computing in a deployment where the T8 Interface is exposed to the IN-CSE, therefore there is a “loose coupling” between the Edge/Fog Node and the Underlying Network.

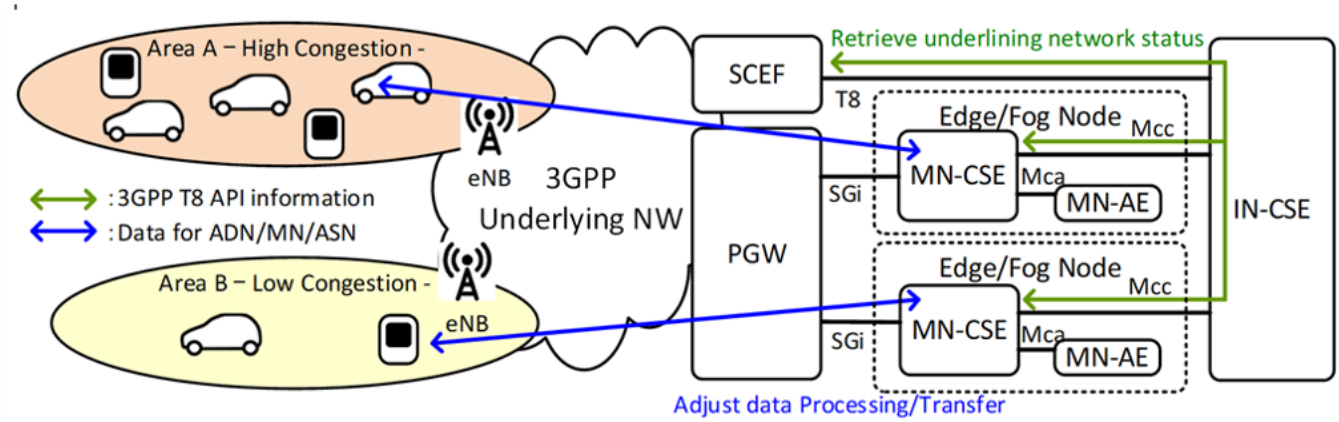


Figure 9.4.2.1-1: High-level illustration - Loosely coupled Edge/Fog computing with 3GPP T8 API

In some Edge/Fog scenarios, an oneM2M Edge/Fog Node can exchange with the 3GPP Underlying Network parameters to be used for optimizing the Data Traffic over the Underlying Network for a set of Field Domain Nodes hosted on UEs. As a result, oneM2M System can avoid the need for the IN-CSE to process Data for the Field Domain Nodes. Figure 9.4.2.11 illustrates the high-level illustration for the loosely coupled Edge/Fog Computing with 3GPP T8 API. The Edge/Fog Node retrieves underlining Network information in a particular area from a SCEF via the IN-CSE and adjusts Data processing /transfer for the Field Domain Nodes.

Artificial Intelligence (AI) and oneM2M IoT Architectures

In Traditional Programming, the Input Data and a Program are fed into a Machine to generate Output. When it comes to Machine Learning (ML), Input Data and Expected Output are fed into the Machine during the Learning Phase, and it works out a Program for itself. To understand this better, refer to the Figure below:

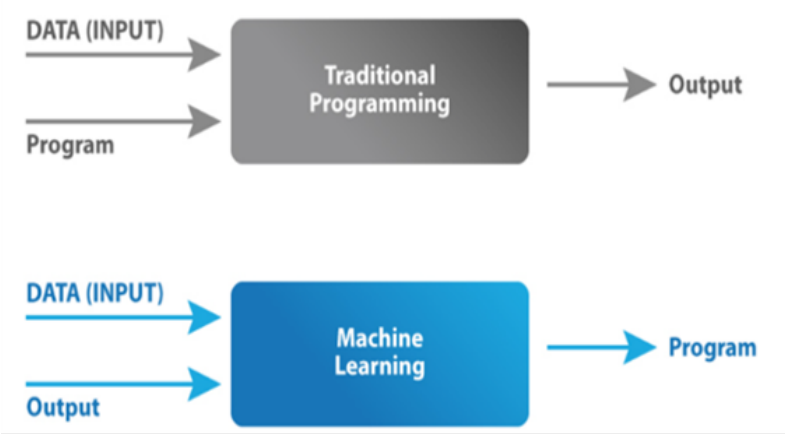


Figure: Traditional Programming vs Machine Learning

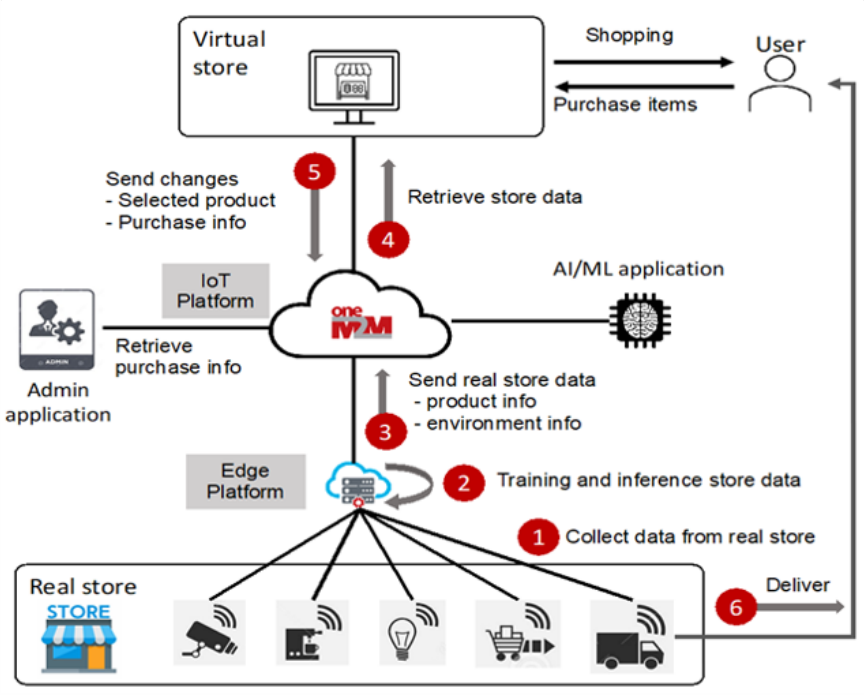


Figure: Conceptual Diagram of Metaverse Virtual Store in oneM2M IoT and Edge Platforms

The word "Metaverse" is made up of the prefix "Meta" (meaning "beyond") and the stem "verse" (a back-formation from "universe"); the term is typically used to describe the Concept of a Future Iteration of the Internet, made up of persistent, shared, 3D virtual spaces linked into a perceived virtual universe. The Metaverse in a broader concept refer to realize Virtual Worlds using IoT, AI and Augmented Reality(AR)/Virtual Reality(VR).

A Metaverse-based Online Store where Stores in the real world are created as Digital Twins in the Metaverse Virtual Space, and Users visit a Virtual Store in the Metaverse Space to purchase preferred products. For Real-Time Synchronization between the Real-World and the Virtual Stores in the Metaverse, various Smart Sensors are used to sense Real-World Products intelligently.

The Edge Node at the Real World store loads a trained AI/ML Model and infers Products' information. The retrieved product Data is then transferred to the IoT Platform for Real-Time Synchronization.

A User can now purchase Products from a Virtual Store in the Meteaverse. The purchase info in the Metaverse is notified to the Administrator and the purchased product is delivered to the user.

The flow of Data has an important bearing on Architectural Components for IoT Solutions, that is part of the analysis of the impact of AI on IoT Architectures, in particular the oneM2M Service Layer. The typical focus is on Data that leads to some form of decision being taken and is classified as 'User-Plane' (UP) Data. The basic Model for IoT Solutions begins with sourcing Data from IoT Devices (illustrated in left-hand side of Figure 1 below). This

Data then passes through a Signal Processing and Machine Learning (ML) Process to extract key features and to represent them as Knowledge-based Objects. The next stage of processing involves the Application of Rules-based AI in areas related to Reasoning, Decision making, Supervision and Explainable AI.

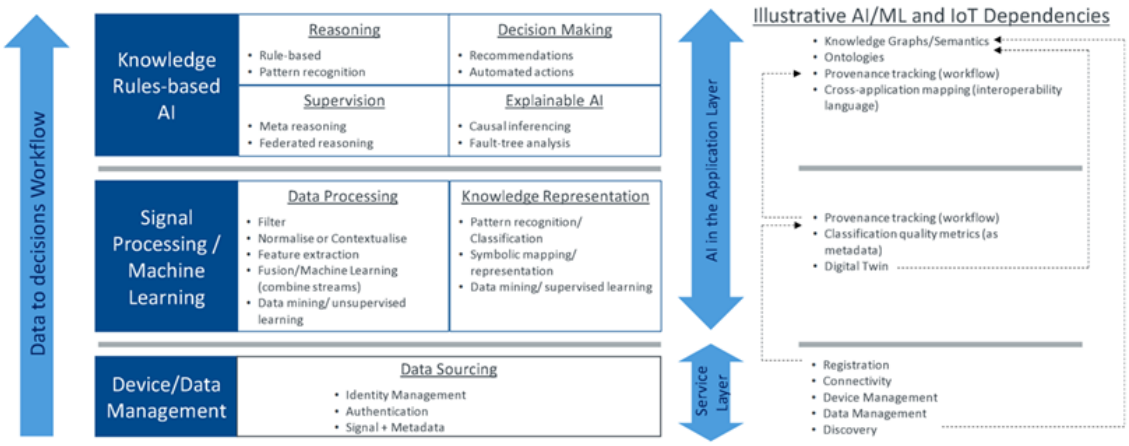


Figure 1: IoT Data to Decisions Workflow

This workflow illustrates some of the Generic and Commonly used Data Sourcing and AI/ML Capabilities involved in supporting End-to-End (E2E) IoT Solutions. It also shows how AI/ML Capabilities depend on Service Layer Capabilities. An example (illustrated in right-hand side of Fig. 1) is the relationship between a **Registration Capability that manages the Identify of a Device and its Value in providing information about the Provenance of Data used in Pattern Recognition or Causal Inferencing Functions.** In this example, the act of tracking data provenance depends on a 'Registration' Service Capability. Provenance tracking can improve the quality and dependability of an AI/ML system and occurs in the background to 'User Plane' (UP) activity. In Architectural terms, such background processes and use of Data that improve the Quality of AI/ML Applications occur in what is referred to as the 'Control Plane' (CP).

A traditional approach (illustrated in Fig 2 below) relies on an Application Ingesting and Storing Data for Processing. The SW implementation concentrates AI/ML and Service Layer Capabilities in the Application Layer. This places a burden on Solution Developers to master Application, AI and Service Layer disciplines.

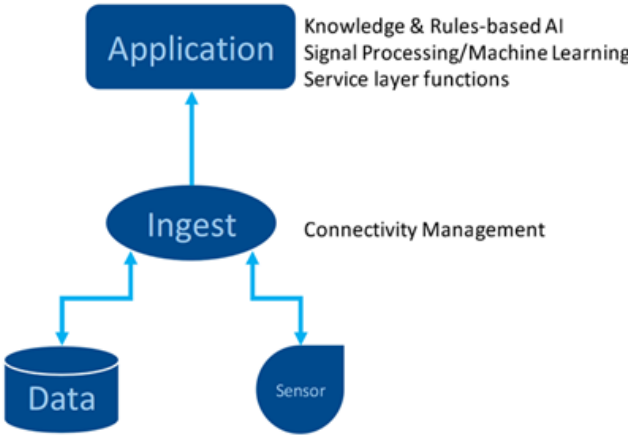


Figure 2: Traditional approach for IoT solutions

The alternative is a Developer friendly Approach (illustrated in Fig 3 below). This approach provides developers with an **Abstraction Layer - i.e. a Commo n Service Layer** - that makes AI and the more usual IoT-enabling Services accessible through a Standardized API. This arrangement means that the IoT Application can rely on notifications from the Common Services Layer to trigger its Functions when notified of changes in IoT Data. **It can also draw on a Library of AI-enabled Services provided within the Common Services Layer.**

A Three-Tier Frramework to organize the logical aspects of AI in IoT maps AI Applications into a 'User' Plane (UP).

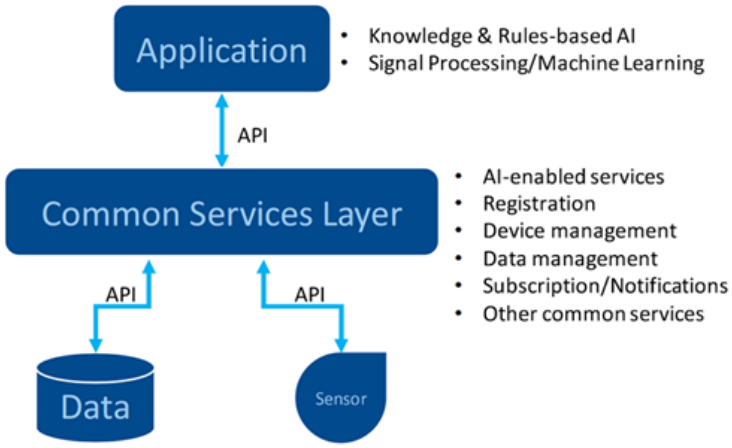
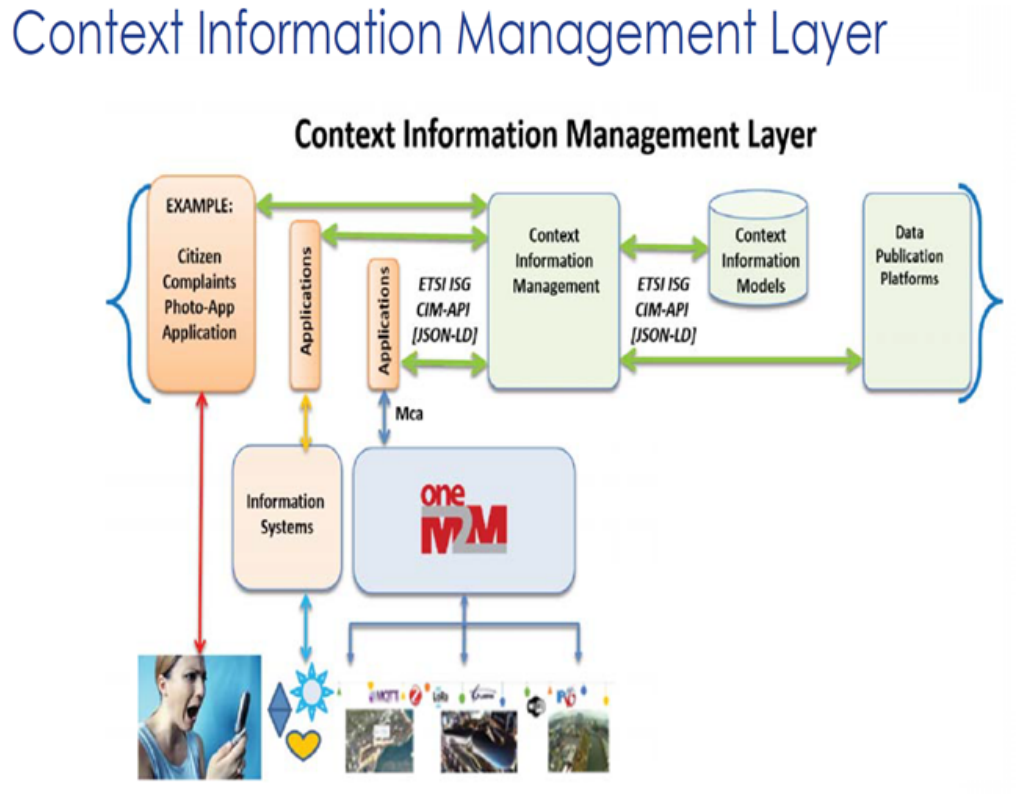


Figure 3: Scalable and developer-friendly approach for IoT solutions

oneM2M and CIM NGSI-LD (Context Information Management Next Generation Service Interface - Linked Data)

The Goal of the ETSI CIM ISG on Context Information Management is to issue TSs to enable multiple Organisations to develop interoperable SW implementations of a cross-cutting Context Information Management (CIM) Layer.



The CIM Layer should enable Applications to Discover, Access, Update and Manage Context information from many different sources, as well as publish it through interoperable Data Publication Platforms.

Phase 1 - detect and describe the Standardization Gaps.

Phase 2 - Developing ISG CIM Group Specifications in Phase 2 will subsequently fill these gaps. It is expected that an extension of the RESTful binding of the OMA NGSI API involving expression using JSON-LD could aid interoperability, so this and potentially other extensions will be considered.

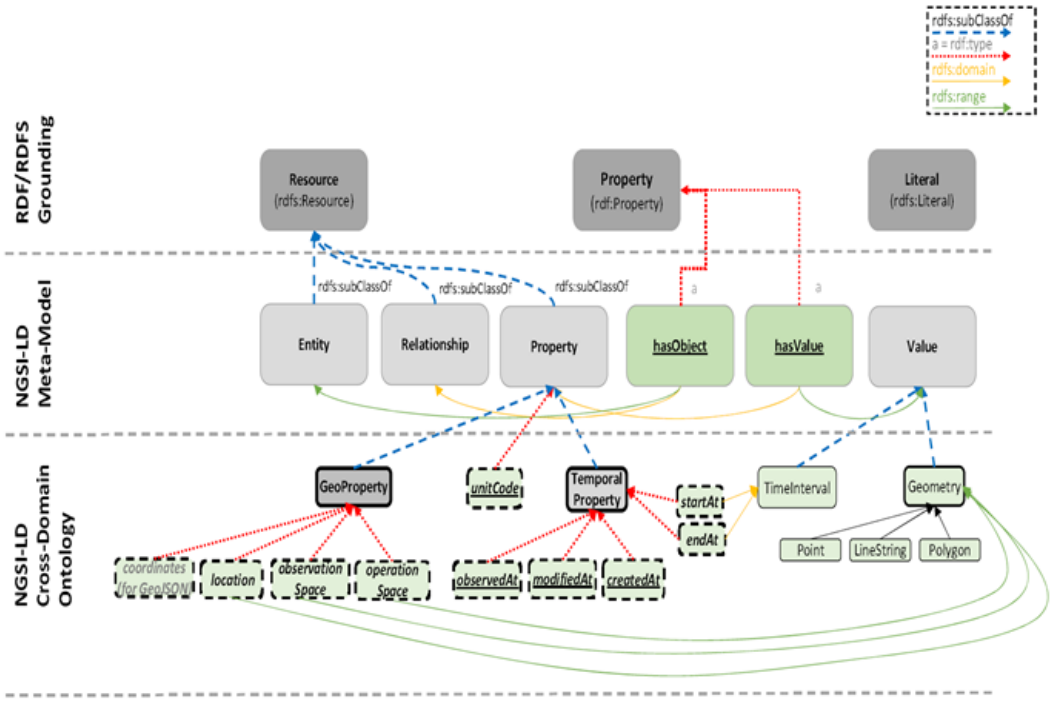


Figure 4.2.3-1: NGSI-LD Core Meta-Model plus the Cross-Domain Ontology

The CIM API allows Users to Provide, Consume and Subscribe to **Context Information c**lose to Real-time Access to Information coming from many different Sources (not only IoT Data Sources).

B.1 Mapping to oneM2M

oneM2M is a partnership project for IoT (originally defined as "machine to machine communication" in the Telecom world). OneM2M provides an OWL ontology that can be partially mapped to the ISG CIM cross-domain ontology, as illustrated in Figure B.1.

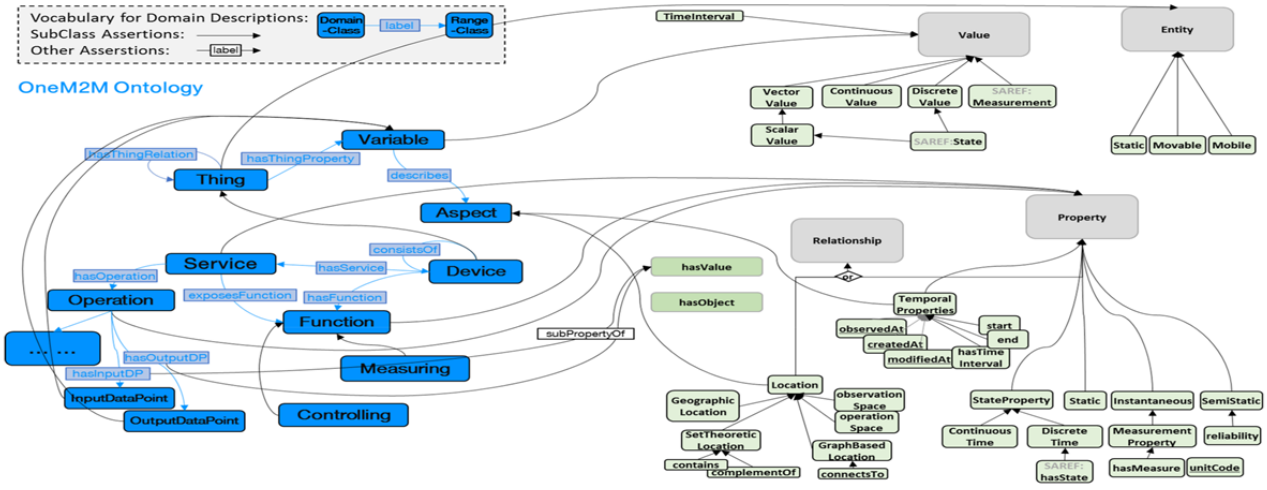


Fig. B. 1: Mapping NGSI - LD Meta - model and Cross-Domain Ontology to oneM2M Base Ontology

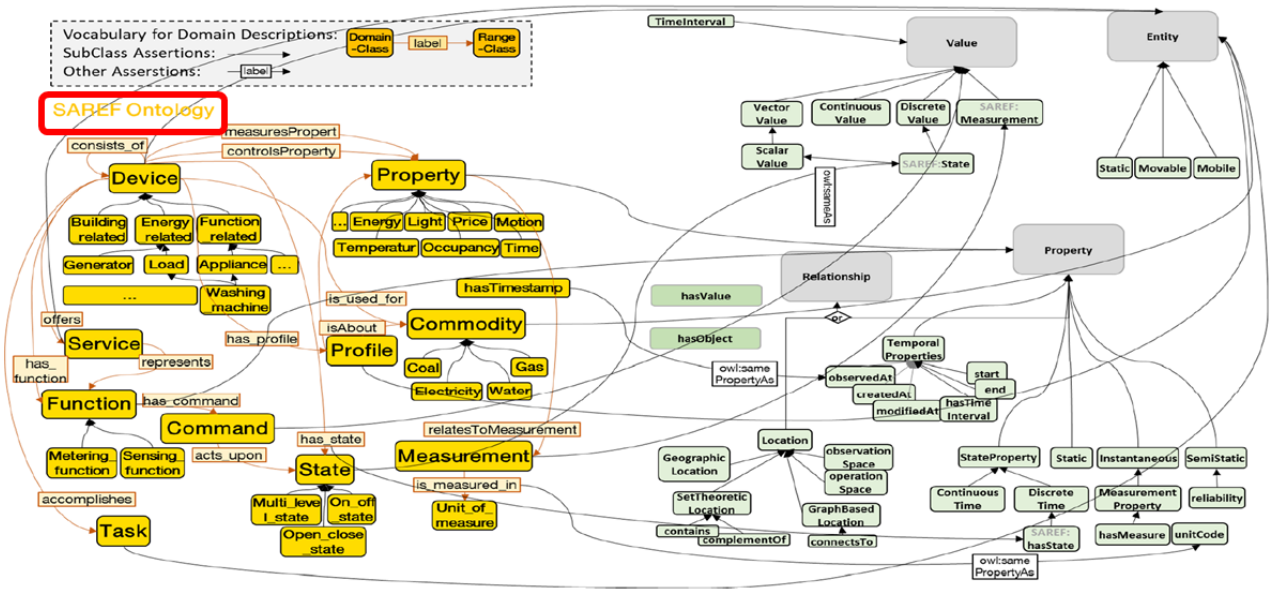


Figure B.4: Mapping NGSI-LD to SAREF

Core NGSI-LD @context

NGSI-LD serialization is based on JSON-LD , a JSON-based format to serialize Linked Data. The @context in JSON-LD is used to expand terms, provided as short hand strings, to concepts, specified as URIs, and vice versa, to compact URIs into terms. The Core NGSI-LD (JSON-LD) @context is defined as a JSON-LD @context which

contains:

- The core terms needed to uniquely represent the key concepts defined by the NGSI-LD Information Model, as mandated by clause 4.2.

- The terms needed to uniquely represent all the members that define the API-related Data Types, as mandated by clauses 5.2 and 5.3.

- A fallback *@vocab* rule to expand or compact user-defined terms to a default URI, in case there is no other possible expansion or compaction as per the current @context.

- The core NGSI-LD @context defines the term "id", which is mapped to "@id", and term "type", which is mapped to "@type". Since @id and @type are what is typically used in JSON-LD, they may also be used in NGSI-LD requests instead of "id" and "type" respectively, wherever this is applicable. In NGSI-LD responses, only "id" and "type" shall be used.

NGSI-LD compliant implementations shall support such Core @context, which shall be implicitly present when processing or generating context information. Furthermore, the Core @context is protected and shall remain immutable and invariant during expansion or compaction of terms. Therefore, and as per the JSON-LD processing rules [2], when processing NGSI-LD content, implementations shall consider the Core @context as if it were in the **last** position of the @context array. Nonetheless, for the sake of compatibility and cleanness, data providers should generate JSON-LD content that conveys the Core @context in the last position.

For the avoidance of doubt, when rendering NGSI-LD Elements, the Core @context **shall always be treated** as if it had been originally placed **in the last position**, so that, if needed, upstream JSON-LD processors can properly expand as NGSI-LD or override the resulting JSON-LD documents provided by API implementations.

The NGSI-LD Core @context is publicly available at https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.3.jsonld and shall contain all the terms as mandated by Annex B.

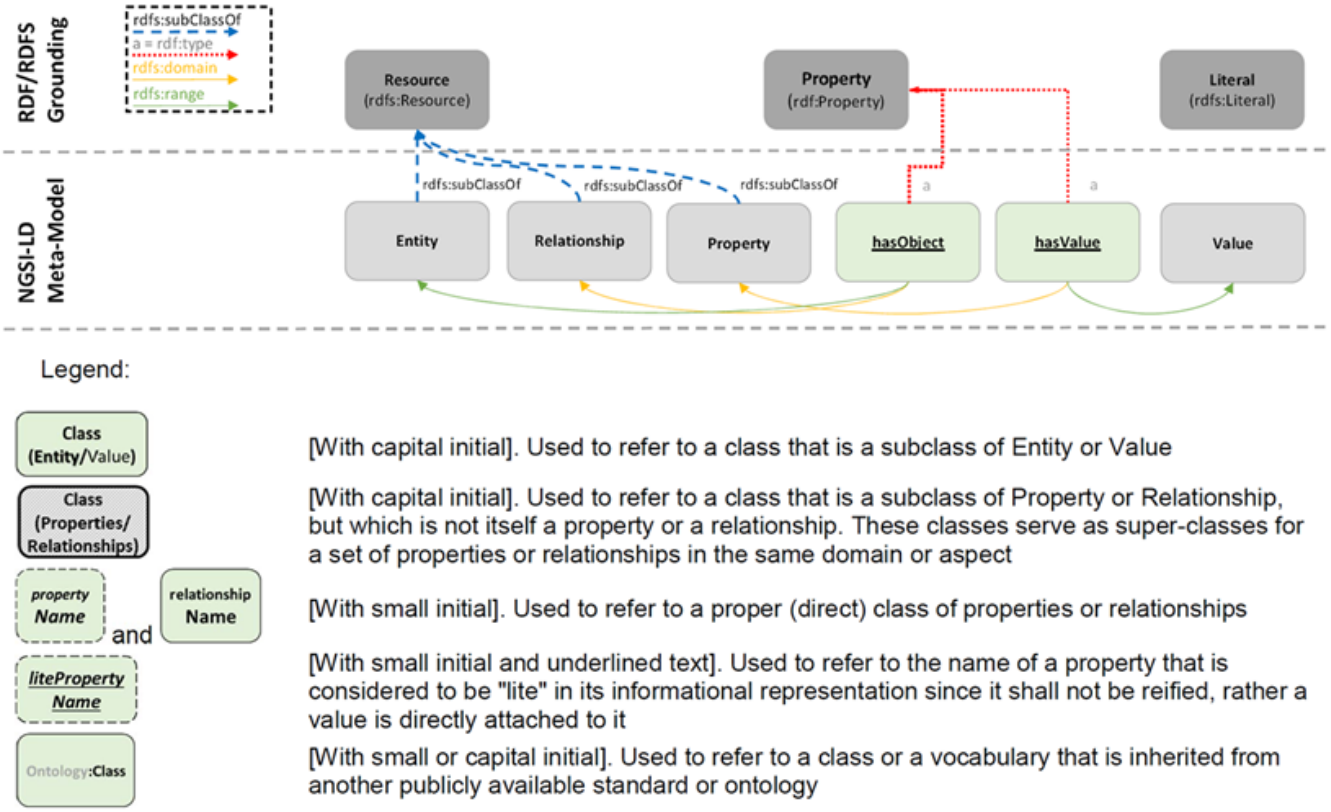


Figure 4.2.2-1: NGSI-LD Core Meta-Model

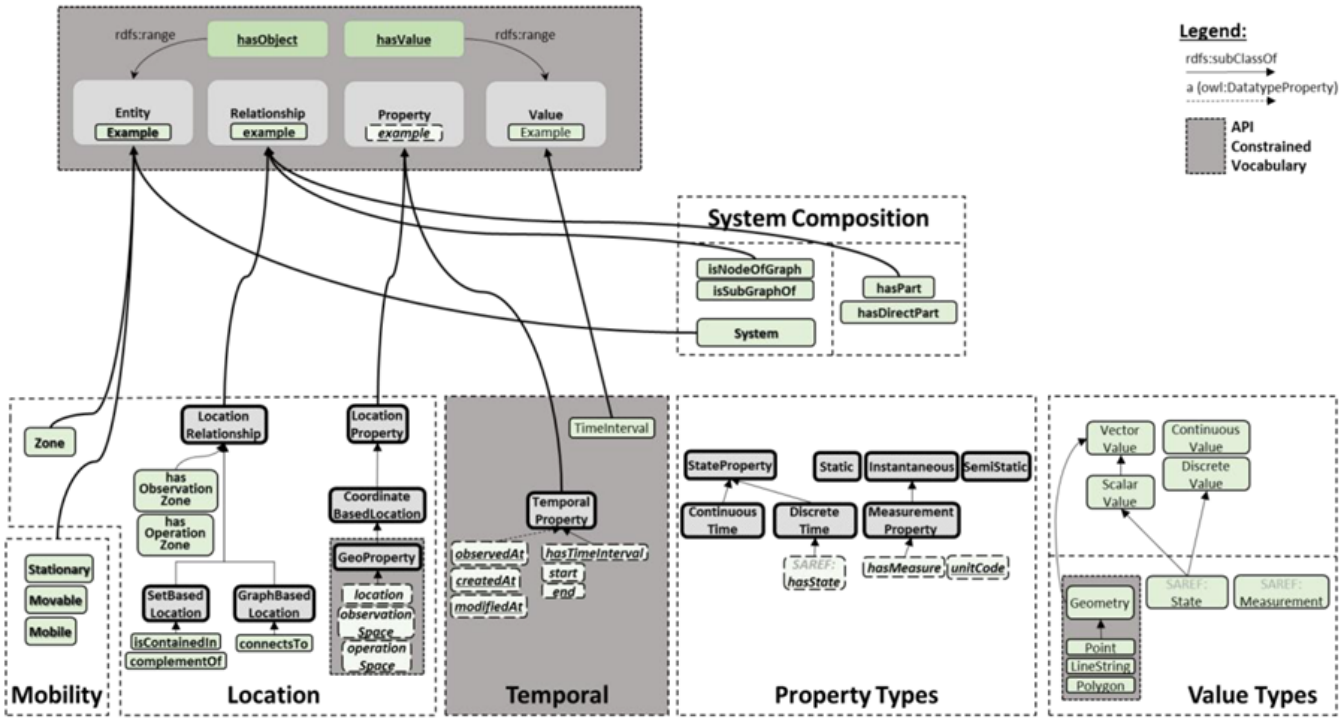


Figure 1 : NGSI-LD Cross-Domain Ontology, with referenced Meta-model[5]

Using Typing vs. using Relationships or Propertiesin ETSI CIM NGSI-LD

Specific types, defined as subclasses of more generic classes, usually have additional properties or relationships that the superclass does not have, or have restrictions different from those of the superclass. There is, however, no single universal criterion to choose between those characteristics of entities that are best expressed by typing, and those that are best expressed by assigning properties.

Typing allows potential checking of data consistency (though full consistency cannot be enforced if external classes are used). Typing avoids replication of pieces of information across all instances of some category of entities that share similar characteristics, precisely because these characteristics may be referenced from the definition of the corresponding super-classes.

Any characteristic feature that is intrinsic to a category of entities, does not vary from one entity instance to another within this category, and may be used to differentiate this category from others, should be assigned to individual instances through typing.

All extrinsic and instance-dependent features should be defined as properties.

For example, the characteristic features of a room defined as a kitchen should set by its type inasmuch as they distinguish it from, say, a bathroom, in a generic way. Its area, and, even more obviously, its state (whether it is empty or occupied) should be defined as a per-instance properties, and whether it is adjacent to the living room should obviously be defined by a per-instance relationship.

Characteristics defined by continuous-valued numbers, or with many possible values such as colours, should be defined through properties and do not normally justify the creation of new classes, except when such a distinction is fundamental to the domain8.

Further modelling choices may be less obvious, for example whether it is useful to define subclasses of the generic kitchen class to e.g. distinguish between open-space vs. traditional kitchens.

It is usually better to use properties than to define sub-classes that might be too specific, too dependent on local cultures, or temporary trends.

In general, using a property with predefined values to capture this kind of subcategorization is not a good idea either (see the clause “Guidelines for Use of Properties” in the following). Yet if a distinction is key to our domain, sub-classing it may also be warranted. If a distinction is important in the domain and we think of the objects with different values for the distinction as different kinds of objects, then we should create a new class for the distinction. A stool would, in this view, not just be a chair with three legs, but a different category of seating furniture altogether. If distinctions within a class form a natural hierarchy, then we should also represent them as sub-classes. If a distinction within a class may be used in another domain (as a restriction, or with multiple typing), then it is also better to define it as a subclass than by using a property.

To summarize: instead of associating similar properties to different entities that belong to a common category, a class can be defined to associate all those attributes9 implicitly to all the instances that belong to it. In case of modification of a property, there would be one local placeholder to change, the attribute associated with the class, instead of changing the attribute in all the instances explicitly.

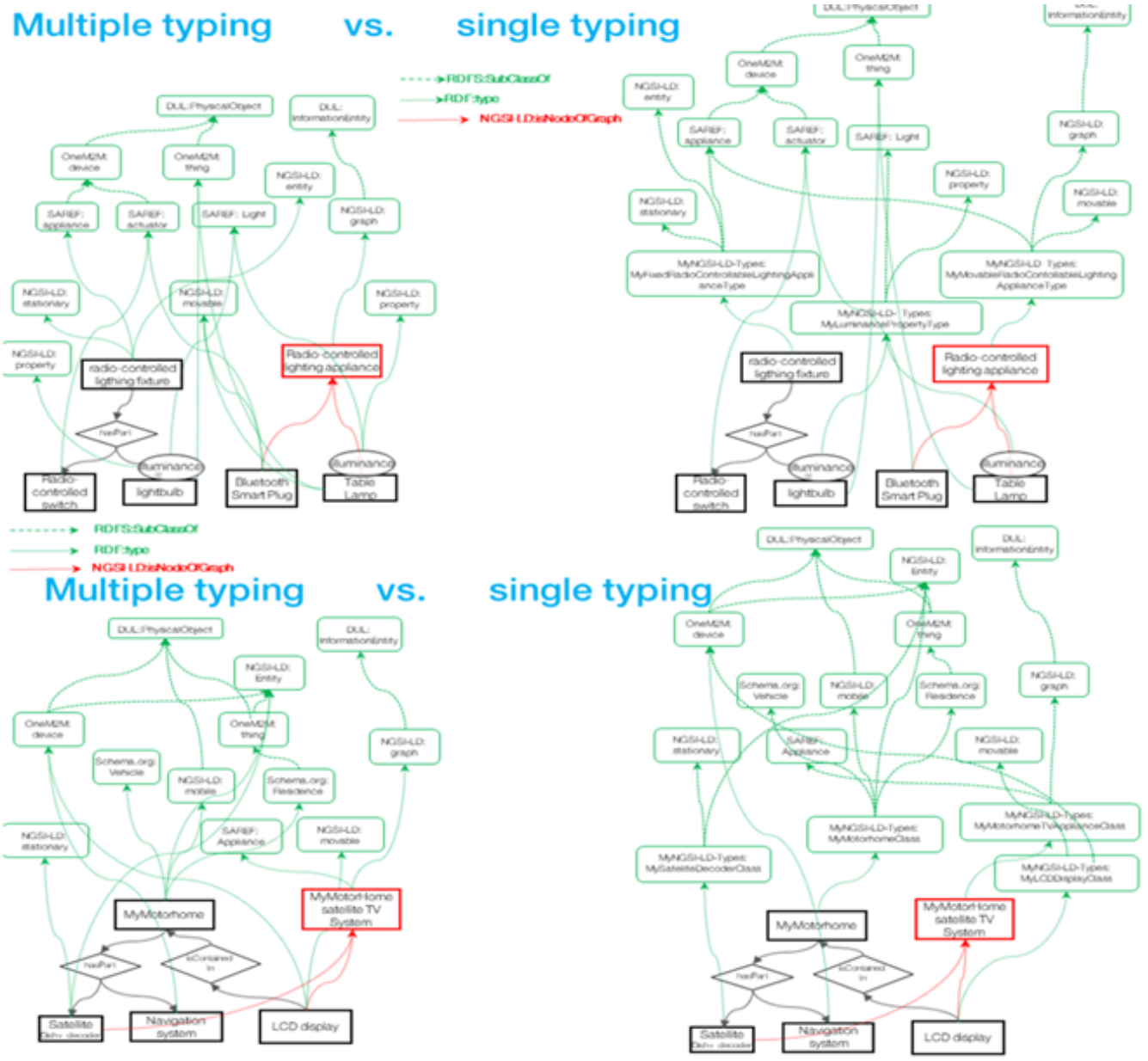


Figure A.1: Examples of different modelling choices depending on the use of multiple typing (on the left) vs. single typing (on the right)

Recommendations for NGSI-LD Typing with Multiple Typing

The following recommendations apply if a form of multiple typing is supported in an NGSI-LD system (which means it does not use the AMPI) and they can be considered for future specifications of the NGSI-LD API.

Contrary to the case of single-typing, multiple typing alleviates the need to create specific "single-typing" classes that exactly match the requirements of the targeted domain and the entity instances to be modelled and may themselves inherit multiple classes from shared ontologies. Instead, with multiple typing, a similar result may be achieved by directly referencing these classes from the instances being categorized, thus picking and choosing known features from a variety of known ontologies, including the NGSI-LD cross-domain and metamodel ontologies. Choices could be made from many ontologies, thesauri, taxonomies, and vocabularies, be they generic or domain-specific, high-level, mid-level or low-level. Classes are not, in general, mutually exclusive, so multiple-typing avoids a granularity that amounts to define every single type by a small, mutually exclusive set of instances, cross-referencing multiple classes being a more versatile and adaptable way to describe their peculiarities than pigeonholing them into narrowly defined categories

With multiple-typing, new instances should be created by :

- referencing, directly or indirectly, at least one of the root classes of the NGSI-LD meta model (entity, relationship, property)

- referencing, directly or indirectly, generic classes from the NGSI-LD cross-domain ontology (for e.g. defining whether the instances being addressed are mobile, movable, or stationary). This is more generic than the use of more specific concepts as they might be defined in domain-specific ontologies.

- referencing multiple classes, chosen from generic or use-case-specific ontologies, to characterize specific features or peculiarities of these instances. With multiple typing, this is preferable to the use of properties or narrowly defined subcategories.

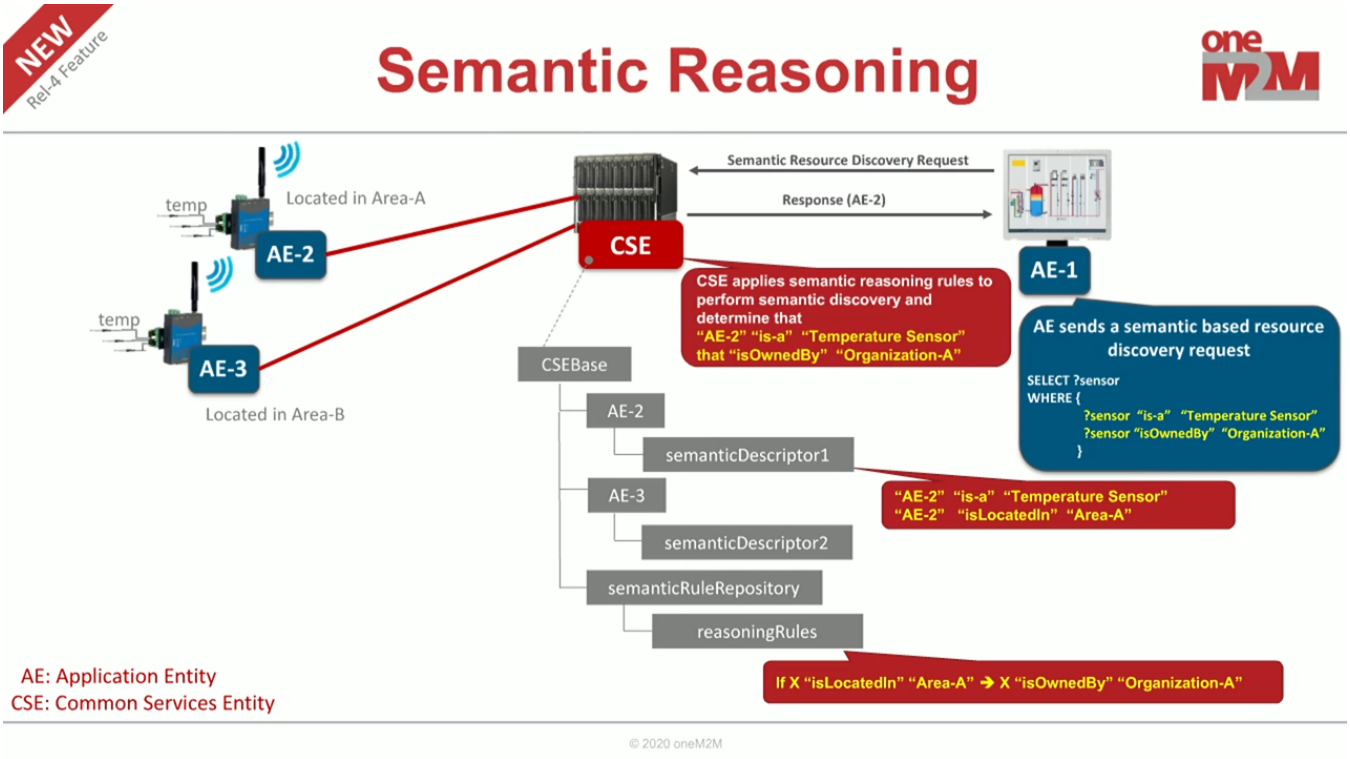
oneM2M Semantic enablement and ASD (Advanced Semantic Discovery) for (AE) "Resources"

The oneM2M Architectural Model in oneM2M Semantic enablement specification is based on the generic oneM2M Architecture for the Common Service Layer specified in oneM2M TS Functional Architecture.

The Core Functionality supporting Semantics resides at various CSEs, providing Services to the AEs via the Mca Reference Point and interacting with other CSEs via the Mcc Reference Point.

The Semantics (SEM) CSF (Common Service Function) is an oneM2M Common Service Function (CSF) which enables Semantic Information Management (SIM) and provides the related functionality based on this Semantic Information. The Functionality of this CSF is based on Semantic descriptions and implemented through the specialized resources and procedures described in oneM2M Semantic Enablement specification

The SEM CSF includes Specialized Functional Blocks such as: SPARQL Engine, Repositories for Ontologies and Semantic Descriptions, which may be implemented via Permanent or Temporary Semantic Graph Stores, etc.



6 Description of Classes and Properties

6.1 Classes

6.1.1 Class: Thing

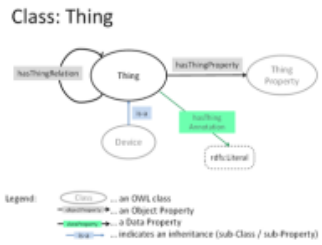


Figure 2: Thing

Description

- A **Thing** in oneM2M (Class: Thing) is an entity that can be identified in the oneM2M System.
  A Thing that is not a Device is not able to communicate electronically with its environment. However, the sub-class of Thing that *is* able to interact electronically is called a "Device".
  A Thing may have ThingProperties (Object Property: hasThingProperty). A Thing can have relations to other things (Object Property: hasThingRelation).
  Since a Thing that is not a Device is not able to communicate electronically it cannot influence the value of its ThingProperties or being influenced by it. Similarly a Thing cannot document its - real-world - relationships (via hasThingRelation) to other Things.

6.1.2 Class: ThingProperty

Figure 3: ThingProperty

Description

- A **ThingProperty** (Class: ThingProperty) denotes a property of a Thing. A ThingProperty can e.g. be observed or influenced by devices, or it constitutes static data about a Thing.
  E.g. the indoor temperature of the room could be a ThingProperty of a Thing "room".
  A ThingProperty of a thing can describe a certain Aspect, e.g. the indoor temperature describes the Aspect "Temperature" that could be measured by a temperature sensor.
  A ThingProperty of a Thing can have meta data.
- The class ThingProperty is a sub-class of the Variable class.

Object Properties

This Class is the domain Class of Object Property:

- describes (range Class: Aspect)
  (inherited from class: Variable)

6.1.3 Class: Aspect

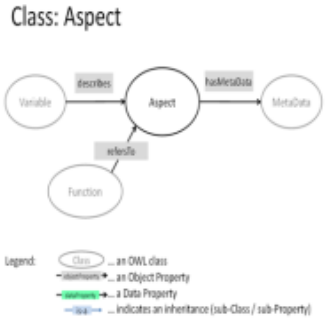


Figure 4: Aspect

Description

- An **Aspect** (Class: Aspect) describes the real-world aspect that a Function relates to. Aspect is also used to describe the quality or kind of a Variable.
  The Aspect could be a (physical or non-physical) entity or it could be a quality.

6.1.4 Class: MetaData

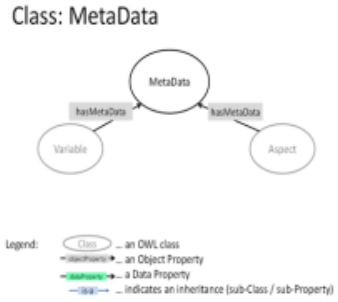


Figure 5: MetaData

Description

- **MetaData** (Class: MetaData) contain data (like units, precision-ranges ....) about a Variable or about an Aspect.
  E.g. the indoor temperature could have as meta data an individual "Celsius_Scale" that specifies that the temperature needs to be understood as degrees Celsius.

Object Properties

6.1.5 Class: Device

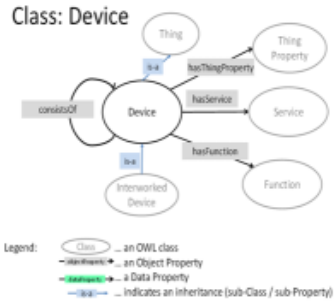


Figure 6: Device

Description

- A **Device** (Class: Device) is a Thing (a sub-class of class:Thing) that is designed to accomplish a particular task via the Functions the Device performs.
  A Device can be able to interact electronically with its environment via a network. A Device contains some logic and is producer and/or consumer of data that are exchanged via its Services with other oneM2M entities (Devices, Things) in the network. A Device may be a physical or non-physical entity.
  A Device interacts through the DataPoints and/or Operations of its Services:

6.1.6 Class: InterworkedDevice

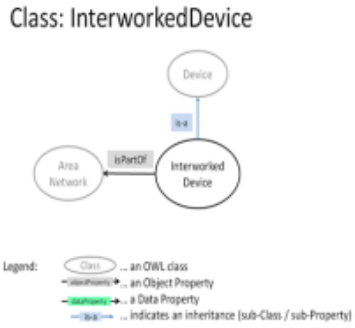


Figure 7: InterworkedDevice

Description

- An **InterworkedDevice** (Class: InterworkedDevice) is a Device - e.g. in an Area Network - that does not support oneM2M interfaces and can only be accessed from the oneM2M System by communicating with a "proxied" (virtual) device that has been created by an Interworking Proxy Entity.

6.1.7 Class: AreaNetwork

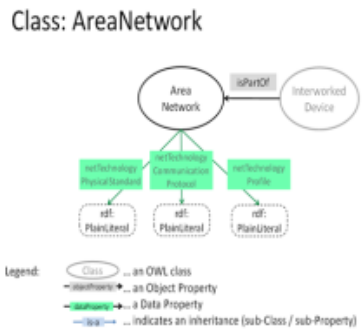


Figure 8: AreaNetwork

Description

- An **AreaNetwork** (Class: AreaNetwork) is a Network that provides data transport services between an Interworked Device and the oneM2M System. Different area Networks can use heterogeneous network technologies that may or may not support IP access.

6.1.8 Class: Service

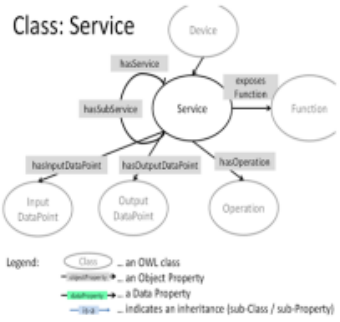


Figure 9: Service

Description

- A **Service** (Class: Service) is an electronic representation of a Function in a network. The Service exposes the Function to the network and makes it discoverable, registerable and remotely controllable in the network.
  A Service is offered by a device that wants (a certain set of) its Functions to be discoverable, registerable, remotely controllable by other devices in the network.
  A Service can expose one or more Functions and a Function can be exposed by one or more Services.

6.1.9 Class: Function

6.1.9.0 General description

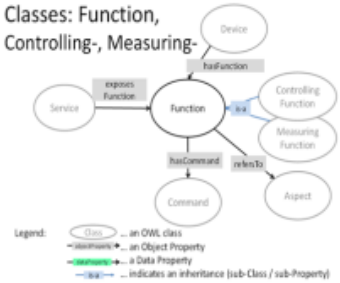


Figure 10: Function

Description

- A **Function** (Class: Function) represents a particular function necessary to accomplish the task for which a Device is designed. A device can be designed to perform more than one Function.
  The Function exhibits the - human understandable - meaning what the device "*does*".
- A Function refers to (e.g. observes or influences) some real-world aspect(s), that can be modelled as a Class: Aspect.

6.1.10 Class: Operation

6.1.10.0 General description

Figure 11: Operation

6.1.16 Class: Variable

6.1.16.0 General description

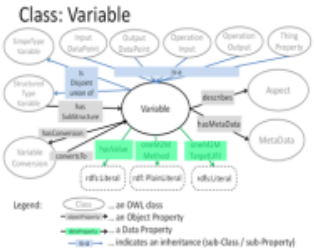


Figure 17: Variable

Description

- A **Variable** (Class: Variable) constitutes a super class to the following classes: ThingProperty, OperationInput, OperationOutput, InputDataPoint, OutputDataPoint. Additionally, class:Variable is the disjoint union of classes: SimpleTypeVariable and StructuredTypeVariable, i.e. any member of class:Variable is also member of either SimpleTypeVariable or StructuredTypeVariable.
  The members of class:Variable are entities that store some data (e.g. integers, text, etc., or structured data) that can change over time. These data of the Variable usually describe some real-world Aspects (e.g. a temperature) and can have MetaData (e.g. units, precision, etc.).

6.1.16.2 Class: StructuredTypeVariable

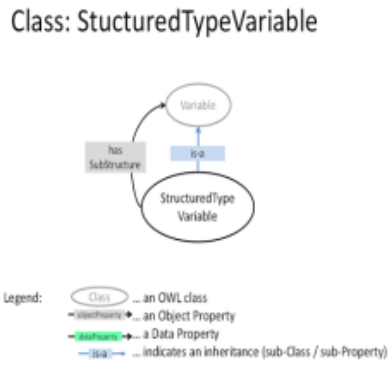


Figure 19: Variable

oneM2M Semantic enablement for (AEs) and ETSI Smart M2M Resource ASD (Advanced Semantic Discovery)

Semantic Discovery in presence of a "Network" of M2M Service Providers (M2MSPs)

The Advanced Semantic Discovery (ASD) aims to discover AEs (also called Resources) that are registered/announced to some CSEs.

The ASD could start from any AE, even these ones not belonging to the same Trusted Domain.

The ASD differs from the usual one present in oneM2M in the sense that one (or many) AE could be searched for even without knowing its identifier, but just knowing its TYPE or ONTOLOGY membership, as shown in Figure 6.3.1-1.

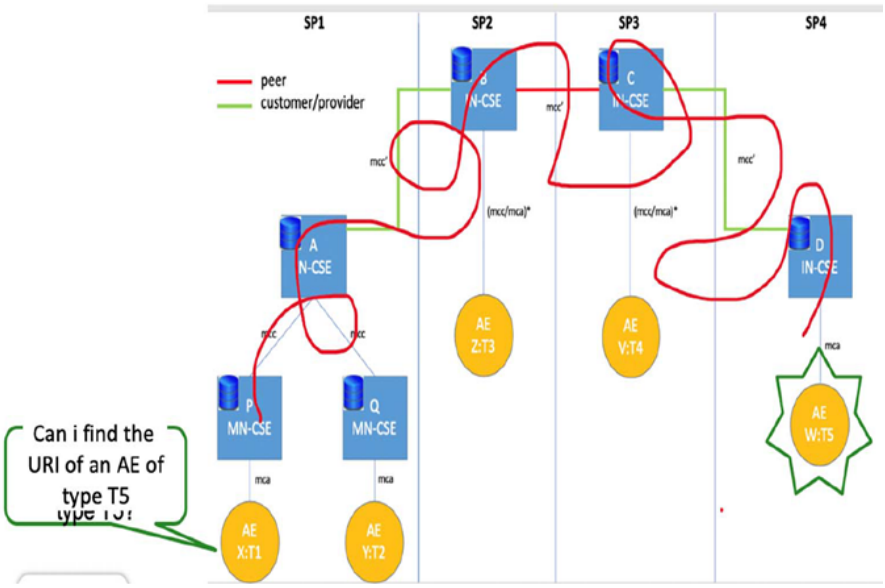


Fig. 6.3.1-1 Advanced Semantic Discovery (ASD) in one image

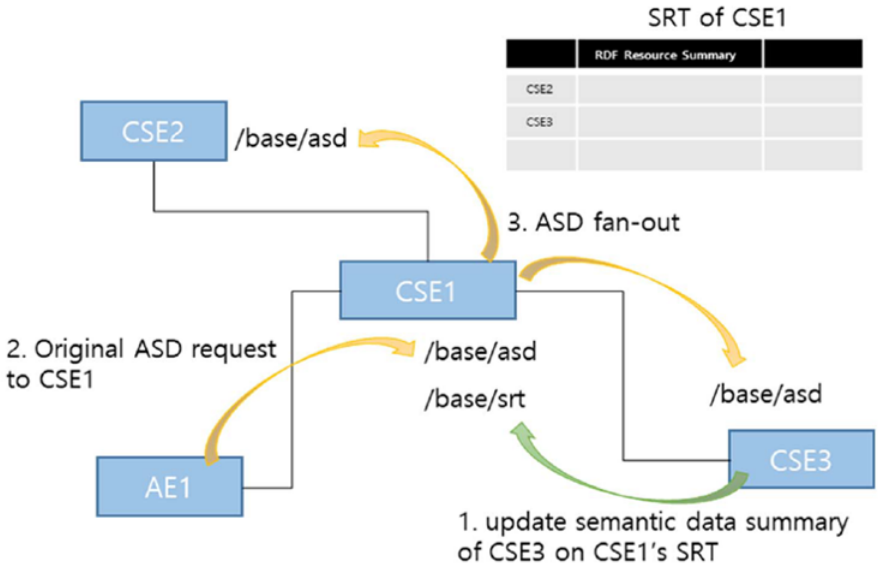


Figure 7.2.1-1: ASD overall illustration

Advanced Semantic Discovery (ASD) in Figure 6.3.2-1 below describes oneM2M as Semantic Discovery involving multiple CSEs.

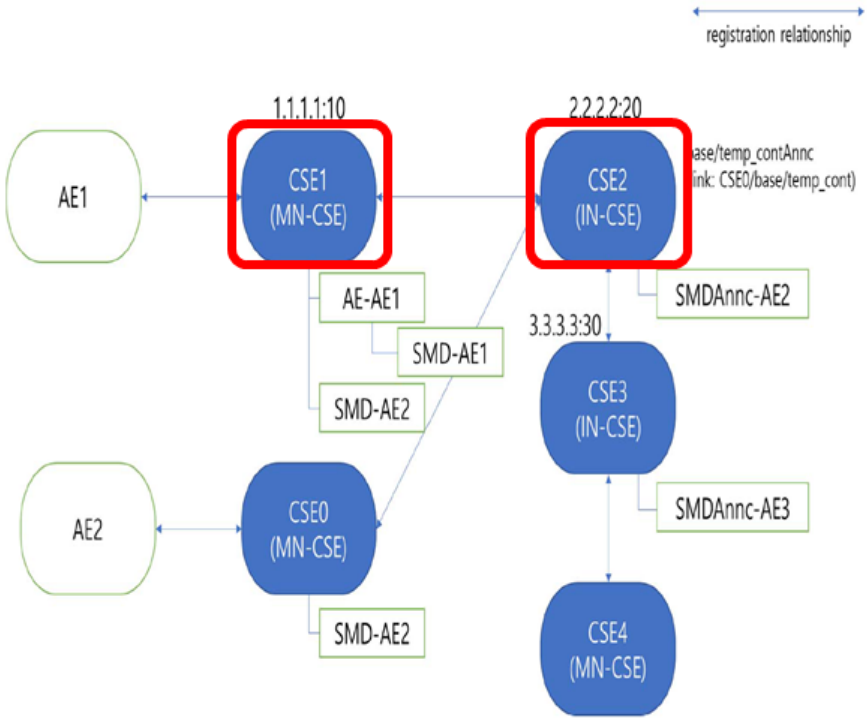


Figure 6.3.2-1: Actual oneM2M limited discovery routing in one slide

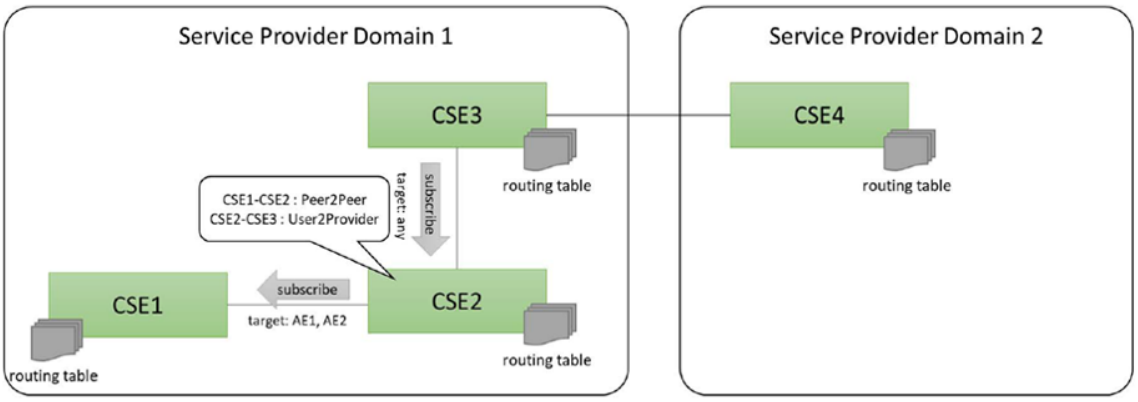


Figure 6.3.2-1: Example of semantic discovery access control

ASD within Distributed Network of CSEs belonging a single Service Provider & across different IoT Service Providers.

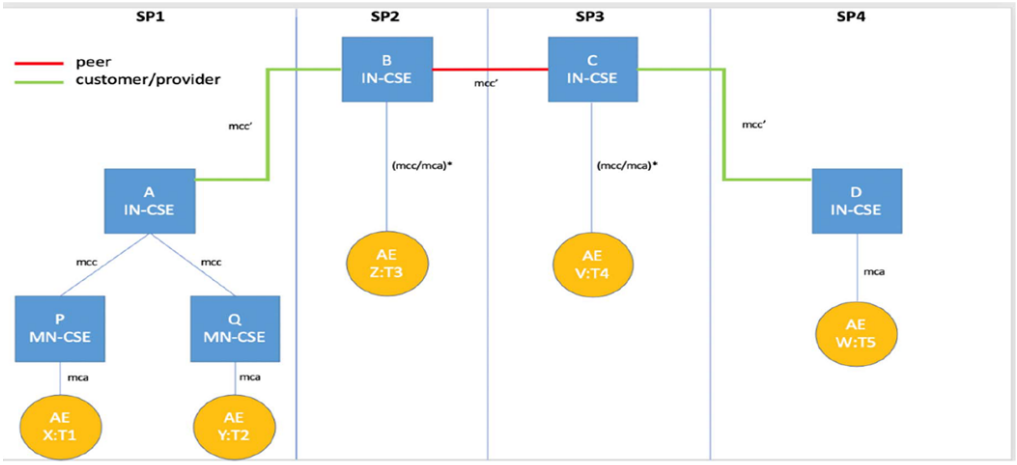


Figure 5.4-1: Pre-condition topology

Example of SRT (Semantic Routing Table)

Table 7.2.6-2: Upgrading the adjacent SRT CSEs with the new y AE-THERMOMETERS

CSEcust1 TYPE	URI	CSE CUSTOMERS	CSE PEERS	CSE PROVIDERS
THERMOMETER	URI_y, URI_w	...	...	(CSE, #_+y)

CSEcust2 TYPE	URI	CSE CUSTOMERS	CSE PEERS	CSE PROVIDERS
THERMOMETER	URI_z	...	...	(CSE, #_+y)

CSEpeer TYPE	URI	CSE CUSTOMERS	CSE PEERS	CSE PROVIDERS
THERMOMETER	URI_a	...	(CSE, #_+y)	...

CSEprov TYPE	URI	CSE CUSTOMERS	CSE PEERS	CSE PROVIDERS
THERMOMETER	URI_b, URI_c, URI_d	(CSE, #_+y)	...	...

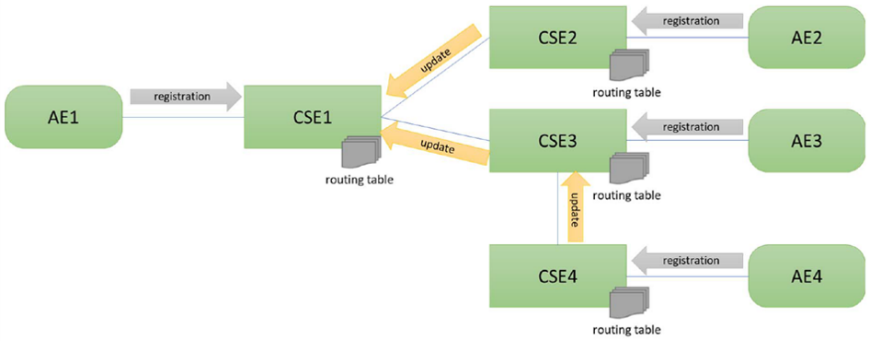


Figure 5.2.1-1: Routing table updates for Advanced Semantic Discovery

The Ontology Mapping Task performed by

=> Create Operation or

=> Update Operation against an <*OntologyMapping*> *R*esource on a Hosting CSE.

A Retrieve operation against the same <*OntologyMapping*> Resource shall be used to get the result of Ontology mapping. A Delete operation against a <*O ntologyMapping*> *R*esource shall follow the basic procedure as specified.

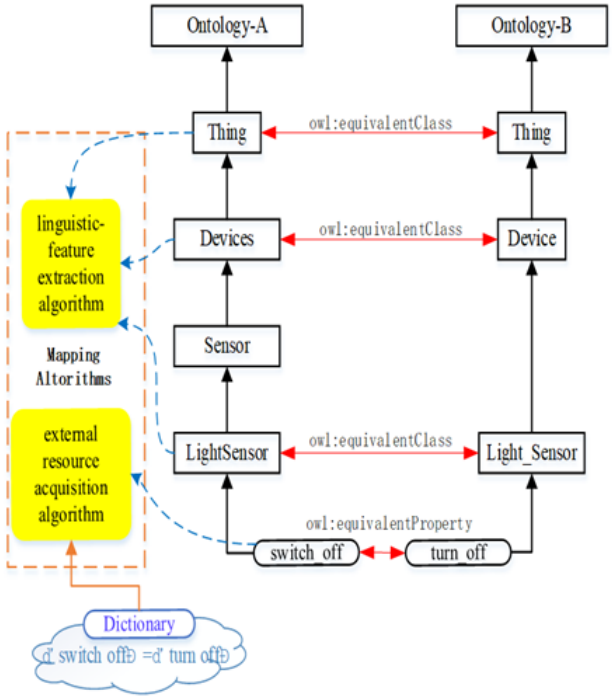


Figure 6.10.2-2: Example of the mapping result between ontology A and ontology B

3GPP 5G NDL (Network Data Layer) and oneM2M Semantic enablement and ETSI SmartM2M ASD integration

The information related to oneM2M Semantic enablement and ETSI SmartM2M ASD support (integration) with 3GPP specified 5G NDL (Network Data Layer in which Data "Compute" is separated from "Storage" in the process of virtualization of 5G NFs into VNFs/PNFs by separating the context in the NF's Application related Data from the Business Logic in the NF's Application related Data and stored separately in Nodes specified by 3GPP for 5G and

denoted as "Structured" and "Unstructured" Data and supported in 5G 3GPP Rel 16 ATSSS (Access Traffic Steering, Switching and Splitting) is deliberately not included and part of the presentation on the oneM2M and 5G New Services.

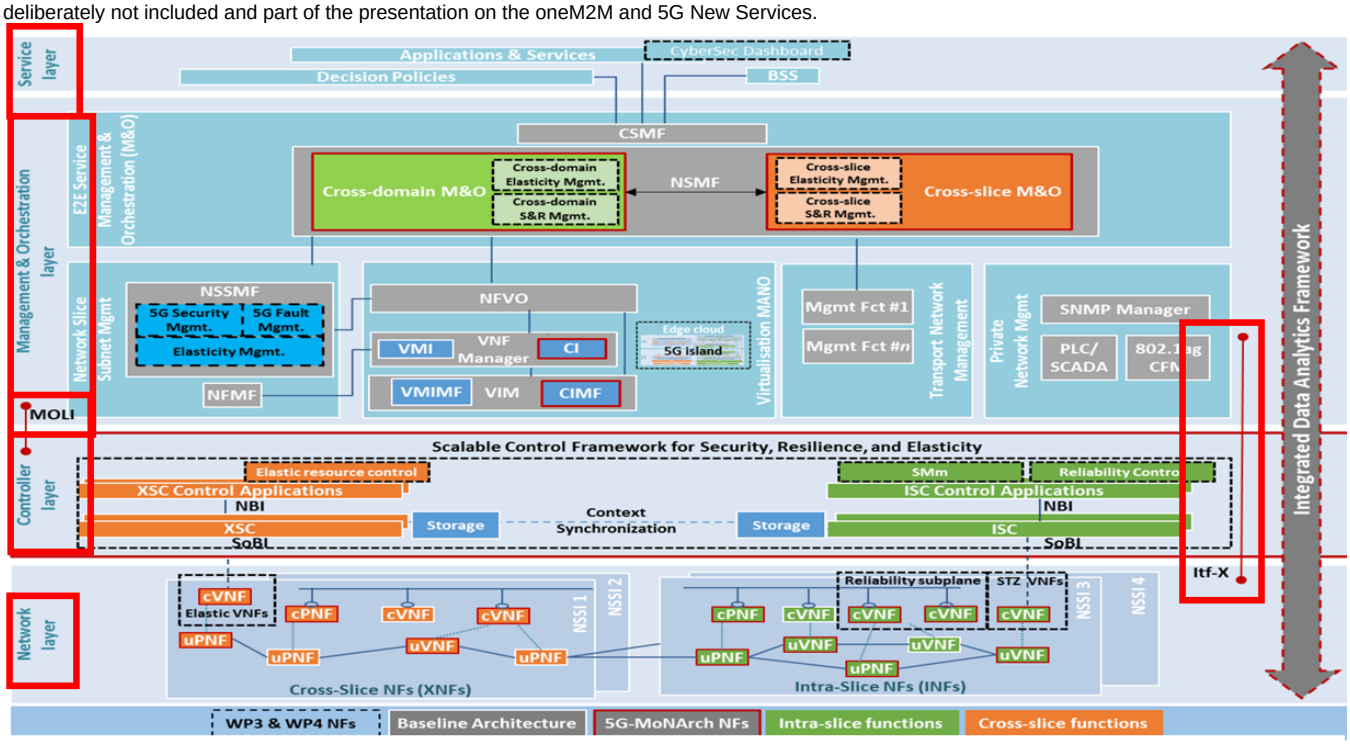


Figure 2-2: 5G Mobile Network overall Architecture

3GPP Release "5G Advanced" System Architecture enhancements for PINs (Personal IoT Networks)

5G System Architecture enhancements for PINs (Personal IoT Networks) in 3GPP 5G Advanced Release

The enhancement of 5G System (5GS) to support Personal IoT Network (PIN) is foreseen to address the Service Requirements for the Personal IoT Networks (PINs) for supporting Management of PIN, Access of PIN via PIN Element with Gateway Capability (PEGC), & Communication of PIN (e.g. PIN Element communicates with other PIN Elements directly or via PEGC or via PEGC & 5GS), enhancements for supporting identifying PIN and the PIN Elements, How to identify PIN & the PIN Elements in the PIN at 5GC level to serve for Authentication/Authorization, Management as well as Policy & Routing Control enforcement. If sidelink is used for the direct communication between PEMC & PEGC, it will be reused the procedures defined for 5G ProSe Direct Communication without introducing new features to sidelink. There shall be no change to underlying non-3GPP Access (e.g. WIFI, Bluetooth) standards. The PEGC & PEMC belongs to same PLMN or (S)NPN.

PINCTRL (*PIN Ctrl Function*) - a new Network Function (**NF**)- & *New Interfaces* P1 & NPINCTF to manage & organize PIN Network as shown in the Figure below. **PEMC/PEGC** communicates with the **PINCTRL of 5G CN** (Core Network) using P1 Reference Points for **Authorization, 3rd Party/Operator Policies** etc. PINCTRL is specified to be part of **SBI bus of the 5G Core Architecture &** communicates with the other **NF(s)** using NPINCTRL Interface. PEMC forms the PIN Network & PEMC/PEGC communicates to the 5G Core on behalf of PIN elements. A **UE** can support both PEGCF & PEMCF & furthermore support the PEF Function in order to **exchange Data Information &**/or provide PIN Services to other PINE in the PIN. The PINE per assumption can use the Non-3GPP Access (e.g. WIFI, Bluetooth) for direct communication to other PINE, PEGC & PEMC & so the following type of Device that contain the PEF are considered: 1) A **Non-3GPP Device,** i.e. a Device that does not support 3GPP Access or **N3GPP Access to 5GC**, but **supports PEF.** e.g. a Device that uses Bluetooth or Wi-Fi communication. 2) A **UE** that supports **N3GPP Access to 5GC** (i.e. **N3IWF**). The UE is restricted to only use the **N3GPP Interface** for PIN direct communication.

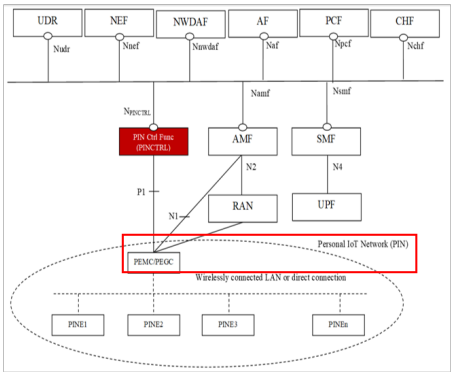


Figure: 5G Core Architecture enhancements to support PIN (Personal IoT Network)

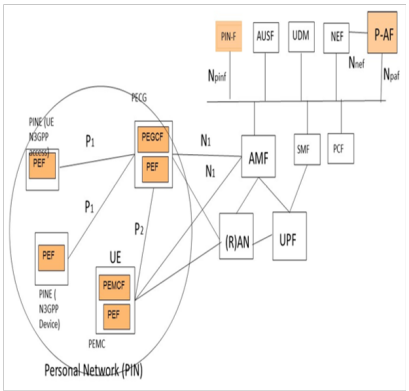


Figure: 5G Core PIN Solution Reference Architecture B

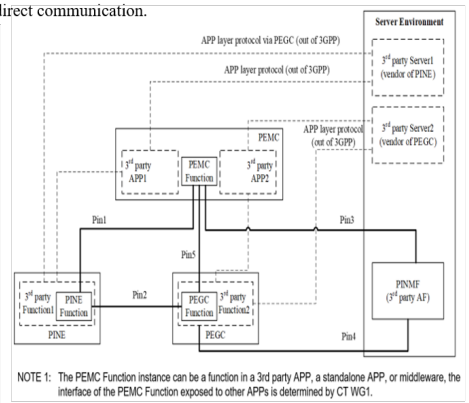


Figure: PIN (Personal IoT Network) Functions deployment example

5G Application Capability for IoT Platforms

The Figure depicts the resulting deployment. Note that this deployment aligns with the distributed network exposure access model introduced by the solution in clause 5.2, while using the proposed IoT-PCS-specific instances of SEAL reference points.

The depicts a generic IoT Platform with IoT Platform Common Services (IoT-PCS) Servers enabling a Set of Applications deployed using corresponding servers (IoT-App), which may belong to different verticals.

On the Device side, corresponding IoT-PCS and IoT-App Clients enable the Client-side functionality.

For Inter-Service Communications, an IoT-App SEAL Server communicates with the IoT-PCS server over the SEAL-X3 Reference Point.

In this deployment, both SEAL Servers provide Network Exposure Access, resulting in a Distributed Network Exposure Access Deployment.

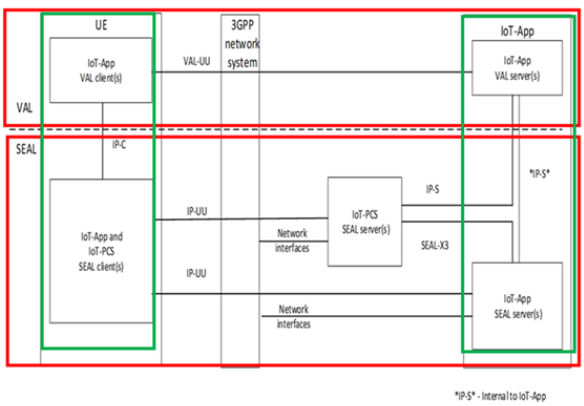


Figure: 5G Functional Model for IoT-PCS (Distributed Network Exposure Access)

1. 5G MSGin5G SBI Service based Interface representation for MSGin5G Service

The MSGin5G, as shown in the figure, is the Service based Architecture for MSGin5G Service.

The M5C Function is the MSGin5G Client.

The AC is the Application Client.

The L3G Function is a Service based function exhibited by Legacy 3GPP Message Gateway.
The N3G function is a Service based Function exhibited by Non-3GPP Message Gateway.

The M5S manages the Distribution of the Messages it has received from MSGin5G UE, from Application Server, or from N3G (on behalf of Non-3GPP UE) or from L3G (on behalf of Legacy 3GPP UE).

The M5S invokes Services provided by L3G/N3G to send MSGin5G Messages towards Legacy 3GPP UE or Non-3GPP UE.

The AS/L3G/N3G invokes Services provided by M5S to send MSGin5G Messages to M5S on behalf of Legacy 3GPP UE or Non-3GPP UE.

The M5S invokes Services provided by SEAL Group Management Function to do MSGin5G Group Management.

The M5S/L3G/N3G invokes Services provided by SEAL Configuration Management Function to do Service Configuration (including UE Service ID Provisioning).

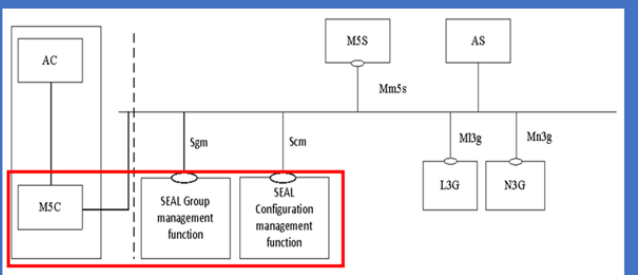


Figure: Service based MSGin5G Architecture

Table: Service based Interfaces supported by MSGin5G Service

Service based interface	Application function entity	Mapping server entity	APIs offered
Mm5s	MSGin5G Server function	MSGin5G Server	Specified in 9.1
MI3g	Legacy 3GPP Message Gateway function	Legacy 3GPP Message Gateway	Specified in 9.2.1
Mn3g	Non-3GPP Message Gateway function	Non-3GPP Message Gateway	Specified in 9.2.2

5G MSGin5G Service overview

The MSGin5G Service is designed and optimized for massive IoT Device Communication including Thing-to-Thing (T2T) Communication and Person-to-Thing (P2T) communication.

The MSGin5G Service is a Message Enabler for applications.

An Application Client in a UE utilizes MSGin5G Service to send a message to another UE, to multiple UEs or to the Application Server, or the Application Server utilizes the MSGin5G Service to send a message to a UE or to multiple UEs. All messages will be routed via the MSGin5G Server in the 5G system. The MSGin5G Service flow is shown in figure 7.1-1.
If the UE supports a legacy 3GPP message service (e.g. SMS, NIDD, or CB) and does not support the MSGin5G Service (i.e. UE has no MSGin5G Client), the message will be translated to the appropriate message delivery mechanism by the Legacy 3GPP Message Gateway. A UE that does not support any 3GPP message service can connect to the MSGin5G Service via Non-3GPP Message Gateway that facilitates the translation between the MSGin5G Service and non-3GPP message delivery mechanism. The connection between such UE and the gateway can be via 3GPP access or non 3GPP access (e.g. WLAN) and is out of scope of the present specification.
An Application Server resides outside the 3GPP domain and connects to the MSGin5G Server via a CAPIF-aware reference point.
The message communication models include:

- Point-to-Point messaging: message that is originated at a UE (UE A) and terminated at another UE (UE B, a Legacy 3GPP UE or a Non-3GPP UE).

- Application-to-Point Messaging: message that is originated at an Application Sever and terminated at a UE.
- Point-to-Application messaging: message that is originated at a UE and terminated at an Application Sever
- Group Messaging: message that is originated at a UE or an Application Server and is terminated at a group of UEs (a group member can be of type UE A, Legacy 3GPP UE or Non-3GPP UE).

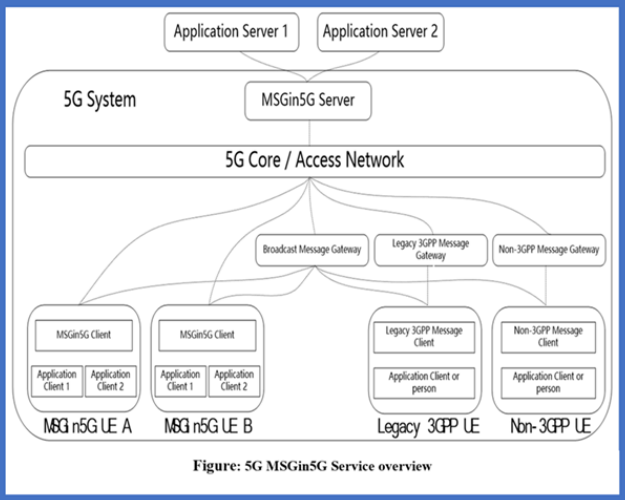


Figure: 5G MSGin5G Service overview

3GPP 5G MSGin5G Application Architecture, Rel. 18, June 2022