ETSI White Paper No. 31

NGSI-LD API:
for Context Information Management

1st edition – January 2019

ISBN No. 979-10-92620-27-6

Authors:

Duncan Bees

Lindsay Frost

Martin Bauer

Mike Fisher

Wenbin Li

ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

About the authors

Duncan Bees

Duncan Bees, Principal, Duncan Bees Technologies Ltd.

Duncan Bees carries out technical and business projects in telecommunications, IoT, media streaming, and broadband infrastructure in Vancouver, Canada. He has led wireless baseband signal processing development teams, product planning for communications semiconductors, strategic planning for the Digital Living Network Association, and was Chief Technology and Business Officer of the Home Gateway Initiative (HGI). He holds the degrees of Master of Electrical Engineering (digital signal processing) from McGill University, and a Bachelor of Applied Science from the University of British Columbia.

Lindsay Frost

Chief Standardization Engineer, NEC Laboratories Europe

Lindsay Frost was elected chairman of ETSI ISG CIM in February 2017, elected to the Board of ETSI in November 2017 and is ETSI delegate to the sub-committee of the EC Multi-Stakeholder Platform (Digitizing European Industry) and to the CEN-CENELEC-ETSI Sector Forum on Smart and Sustainable Cities and Communities. He began his career as a research manager in experimental physics facilities in Germany, Italy and Australia, before joining NEC in 1999. From 2003 to 2009 he managed NEC R&D teams for 3GPP, WiMAX, fixed-mobile convergence and WLAN, while also working for two years as a group chairman in the Wi-Fi Alliance.

Martin Bauer

Senior Researcher, NEC Laboratories Europe

Martin Bauer has worked on activities related to Internet of Things, Context Management and Semantics for more than 15 years. Recently he has been working on IoT-related European projects in the area of smart city and autonomous driving, as well as IoT-related standardization activities, in particular oneM2M and ETSI ISG CIM. He is the AIOTI WG03 sub-group chair for the semantic interoperability topic. He holds a doctorate degree in Computer Science from Stuttgart University and Master of Science degrees in Computer Science from both the University of Oregon and Stuttgart University.

Mike Fisher

Chief Researcher of Distributed Computing, BT Applied Research

Mike Fisher's main research interests are in the convergence of computing and wide area networking - including active networks, Grid computing, Cloud computing and most recently the Internet of Things. He has had a leading role in BT's contributions to a number of collaborative IoT projects, including the MK:Smart and CityVerve smart cities projects in Milton Keynes and Manchester, respectively. His focus in BT was on network services to make information sharing easier, and the value that this can deliver.

Wenbin Li

Research Engineer, Easy Global Market

Wenbin Li is a Research Engineer at Easy Global Market working on international projects (including FP7, H2020) in IoT and ICT domains. The research fields cover IoT architecture and platform, data modelling and management, Semantic Web, and AI. He received a PhD degree of Computer Science from the National Institute of Applied Science Lyon with a special focus on services-oriented computing paradigm in dynamic environments. Before joining Easy Global Market, he was a R&D engineer of Internet of Things at Orange Labs, during which he contributed to the European project FP7 FICORE and an Orange internal project to improve semantic interoperability for IoT infrastructures.

Contents

Introduction: NGSI-LD and Linked Data

The ETSI Industry Specification Group for cross-cutting Context Information Management (ISG CIM) named its API using the text string "NGSI-LD" with the formal agreement of OMA which originally defined NGSI. This greatly helps in avoiding confusion with the IEC CIM/Common Information Model specifications. The rationale is to reinforce the fact that it leverages on the former OMA **NGSI** 9 and 10 interfaces [3] and FIWARE NGSIv2 [4] to incorporate the latest advances from **Linked Data**.

This whitepaper explains the main concepts behind a new data exchange protocol called NGSI-LD which aims to make it easier to find and exchange information with open databases, mobile Apps and IoT platforms. It fills the gap between brief press releases and detailed specification documents for NGSI-LD API [1] and related use cases [2].

New applications in domains such as Smart City, Smart Agriculture, and Smart Industry collate and interpret data from fast-changing and geographically dispersed sources. They also actuate devices and effect many real-world results. To produce, interpret and exchange data, these applications need to define unambiguously the data used, and to share those definitions with other applications. The data relevant to a service, and the definitions that describe its format and meaning, can be called the *context* of the service. For example, location, time, temperature, and application specific information must have common definitions and be understood by all the applications which manipulate it.

In domains like Smart City, the lack of an open and standardized approach for the exchange of context information is a hindrance to the widespread adoption of services. Consider that a dramatically growing number of entities – such as an Internet-of-Things (IoT) enabled traffic sensor, traffic light, or a vehicle license database – can provide information to, receive information from, or receive control messages from many different applications. A common framework for context would enable re-use and information sharing across those applications and help build a critical mass of rich services. This is illustrated in the figure 1, where a common context definition for multiple services (on the right of the figure) is contrasted with the inflexibility of independently defined context (on the left).

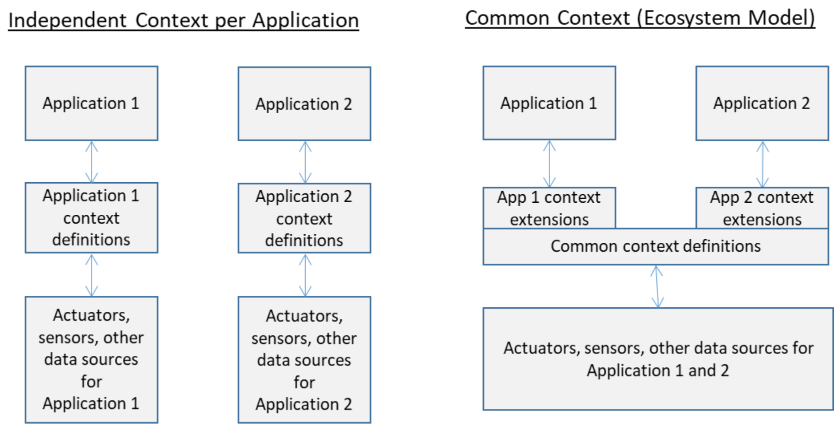
Independent Context per Application
Application 1
Application 2
Application 1 context definitions
Application 2 context definitions
Actuators, sensors, other data sources for Application 1
Actuators, sensors, other data sources for Application 2
Common Context (Ecosystem Model)
Application 1
Application 2
App 1 context extensions
App 2 context extensions
Common context definitions
Actuators, sensors, other data sources for Application 1 and 2

Figure 1: Independent Context vs Common Context (Ecoystem Model)

The ETSI Industry Specification Group for cross-cutting Context Information Management is addressing this problem. This whitepaper describes an open framework for the exchange of contextual information for smart services, using a RESTful[1] API named NGSI-LD, developed by ETSI ISG CIM. The term NGSI points to earlier work by the Open Mobile Alliance[2] which was one inspiration for our approach and the term LD points to concepts of Linked Data as the other major influence.

Context information exchange using NGSI-LD has major advantages. Firstly, within the NGSI-LD framework, applications can flexibly discover and query relevant information. The data discovery is dynamic, and the built-in query patterns support the most common questions that are practical in unbounded federated information systems. Secondly, NGSI-LD helps to precisely communicate the nature of the context information for a given service, such as its period of validity, its geographic constraints, and other semantically important information, by enabling direct inclusion of pointers to the relevant parameters and definitions. To ensure interoperability, the NGSI-LD API defines the meaning of the most commonly needed terms and provides the tools to create domain-specific extensions to model any other type of information. Thirdly, NGSI-LD provides a scalable solution to connect, publish and federate diverse data sources using a developer-friendly interface for data sharing and usage.

More specifically, the data in NGSI-LD are structured like a data graph model and linked with each other. The relationships between entities are easily handled in NGSI-LD in a similar way to graph databases. It also provides the update, query and subscription support needed to allow applications to automatically access data from various data sources. The deployment architectures, examples of which are described later in this paper, are also flexible.

NGSI-LD builds upon previous work, including OMA [3] and FIWARE [4]. It is intended to complement and work alongside IoT system specifications, while providing a path to integrate data from open linked data and other information sources. In NGSI-LD, information not relevant to the application layer of that particular service, for example details of the IoT or network technologies used to connect entities, or to manage IoT devices and gateways, is not explicitly considered. Such aspects are covered elsewhere, for example in oneM2M, where ETSI is a founding member alongside other major international standards bodies. OneM2M has defined a common platform for IoT, which can interoperate with a wide range of networks and systems (oneM2M, TS-0001-Functional_Architecture [5]).

The NGSI-LD information model makes it easy to create models of real-world entities, relationships and properties; moreover, the information model is expressive enough to connect and federate other existing information models, using JSON-LD [6]. It is also compatible with RDF so that triplestores and application logic found e.g. in SPARQL[3] or DataCube[4] software can be applied.

The ETSI ISG CIM work is ongoing and interested parties are invited to participate. Information useful to implementers, as well as details about the group, are provided towards the end of this paper.

[1] The term RESTful (representational state transfer) was introduced in 2000 by Roy Fielding to refer to a simple and highly scaleable way of accessing and manipulate textual representations of web resources using a uniform and predefined set of stateless operations, usually the HTTP operations GET, POST, PUT, DELETE.
[2] OMA NGSI v9 and v10, see www.openmobilealliance.com and successor organization www.omaspecworks.org
[3] SPARQL https://www.w3.org/TR/sparql11-query/
[4] DATA CUBE https://www.w3.org/TR/vocab-data-cube/

Context Information

We can depict a “smart” service as an interconnection of context providing services and context consuming applications. These work together to ensure that each application has the information it requires to deliver knowledge and insight, and to exercise control. We can regard the context of an application as being all the relevant aspects of its operating environment that are required for it to work as intended. Each application needs a different mix of data (context) from one or more sources. A context producer may be a sensor, a gauge, a database an open data repository, etc.

In figure 2, context producers and consumers are connected by a cross-connecting Context Information Management System. The NGSI-LD API is used by data consumers to query for and receive updates on *context information*.

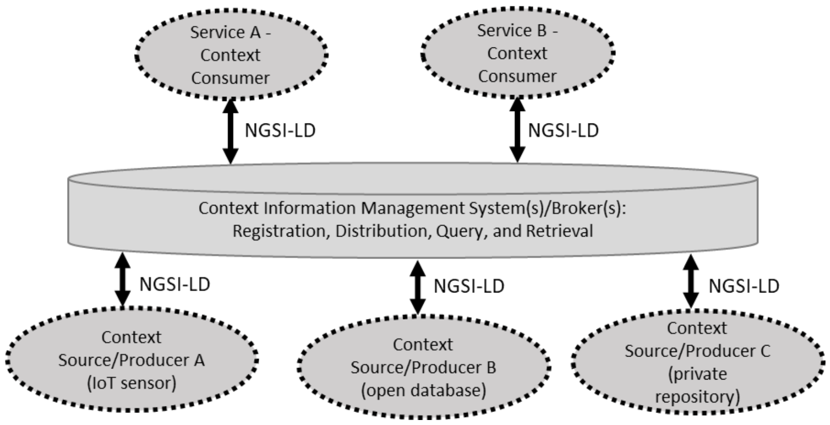
Service A - Context Consumer
Service B - Context Consumer
NGSI-LD
NGSI-LD
Context Information Management System(s)/Broker(s): Registration, Distribution, Query, and Retrieval
NGSI-LD
NGSI-LD
NGSI-LD
Context Source/Producer A (IoT sensor)
Context Source/Producer B (open database)
Context Source/Producer C (private repository)

Figure 2. Interconnection of Context Information

In the ETSI ISG CIM framework, context information is considered to be any relevant information about entities, their properties (temperature, location, or any other such parameter), and their relationships with other entities. Entities may be representations of real-word objects but may also be more abstract notions such as a legal entity, corporation, nation state, or groups of entities.

For example, a smart electric meter may be modelled as an entity of a defined type, installed in a house at a given location, measuring a dynamically changing power consumption, and connected to a particular distribution transformer. A Smart Energy application may require all this information as context.

An important feature of NGSI-LD is that also the relationships between entities can be modelled as having specific properties, e.g. Anne and Bob have a relationship "isMarriedTo" and this relationship has a property called "DateOfMarriage". Information models which provide for annotating attributes to relationships are commonly called labelled property graphs (see references [7] and [8]). They provide a theoretical foundation for automated reasoning about the characteristics of the systems they represent.

Context information is exchanged amongst applications, context producers, and context brokers. The key requirements driving the technology choices for NGSI-LD and then the Architectures are discussed below.

Requirements driving the NGSI-LD Solution

The main considerations shaping the design of the NGSI-LD API are given here, and in more detail below.

- The information model and API should model context information such as entities, their properties, and relations. Communication protocols and other IoT system parameters are not explicitly modelled (unless relevant to the service context, for example including an ID of a wireless access point as part of the provenance of a value measured by a sensor).
- Context producers should be able to register and update the broad categories of information they can offer. Context consumers (software applications and ultimately end users) should be able to discover relevant context information and receive notifications of updates.
- Flexible query options should be supported. An application may need, for example, to query all parking facilities in a city, check which ones have charging points for electric vehicles of the desired type/power-rating and check availability.
- Commonly needed cross-domain constructs such as time and location should be explicitly defined in the information model, to prevent minor variations leading to system incompatibilities. The information model should be extensible so as to allow the creation of domain/application specific definitions and semantics.
- A range of situations and architectures should be supported from the simple to the very complex systems with huge numbers of entities. Deployed architecture should be able to evolve, from centralized to distributed to federated, without needing to reinstall software implementations.

Data Lakes, Context Brokers and the NGSI-LD Architecture

There is nothing new about databases exchanging information. In the last years the need to organize information within city administrations and also across globally distributed enterprises has led to the development of so-called “data lakes” where a (very) large storage system, with indexing appropriate to the applications and use cases, is connected with a large number of different databases with varied data structures and varied data definitions. So-called “adaptor” software is built, often unique to the system, to import the information from the various sources, adapting the data and matching the terms used to categorize it, so that the imported information is consistent with the definitions and indexing defined for the “data lake” i.e. so-called data cleanup. The importing can be dynamic, as information is added or changed in the contributing databases. Cleanup can include consistency checks, timestamp validations, etc. Nobody wants a dirty data lake.

NGSI-LD API can add value also to these sophisticated proprietary systems by providing a system-independent means to advertise the existence of data with particular categories and provide it using an appropriate adaptor. Systems outside the ecosystem of a specific data lake can use the API to extract just the information they need. It is important to see the API as a “messenger” which is independent of specific data sources and data lakes, as well as independent of specific information brokers and architectures.

NGSI-LD System Elements and Architectures

What are the ways in which the NGSI-LD API can be used within a real-world service architecture? In the ISG CIM framework, context consumers use the API to access information provided by context producers, often through *Context Brokers* of several types. A *Context Broker* mediates exchanges between context consumers and producers. The architectures shown in the following are prototypical ones; real systems may instantiate a combination of these architectures or form a more complex one, e.g. consisting of a mesh of Context Brokers.

A *Context Producer* (e.g. a sensor gateway, which sends streams of updates or timed batches of updates) simply delivers context information to context brokers, which arrange its storage and (later) delivery to context consumers.

A C*ontext Source* is the name given to more complex systems which can collect and store a wide range of information, e.g. from devices or databases, register the availability of its information with a broker's *Context Registry*, and deliver the requested context information in response to a redirected query from a Broker or directly to context consumers.

The functionality of a Context Source can of course be paired with a Context Broker. This is a common architecture, to provide a *Central Broker* which acts as the central point of context management and storage for many context producers, as shown in figure 3. This architecture might be chosen for a system of limited geographical scope, or one where the end devices are limited in capability and storage. For example, a context producer could be a rain sensor which is connected via an IoT system to a Central Broker. Context Producers use update operations to update the context information stored in the Central Broker. Context queries to the Central Broker from context consumers could be either one-time or could be asynchronous subscribe/notify queries. Note that a component could be both a Context Consumer and a Context Producer e.g. an App in a mobile phone might query about air pollution conditions along a cycling route or also provide its location on the route to enable statistics about the popularity of bicycle routes to be collected.

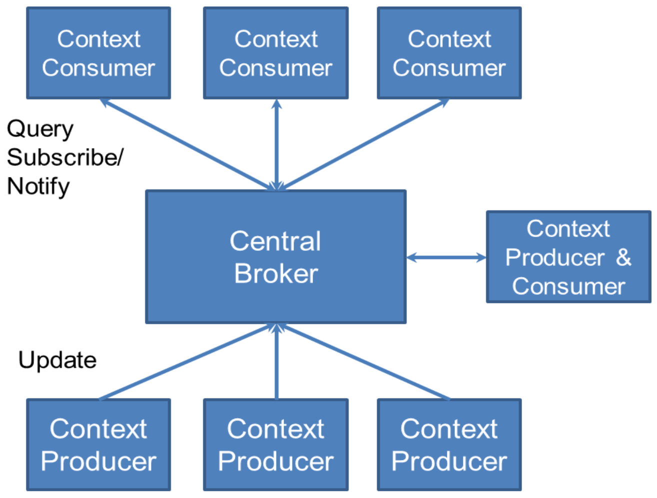

Figure 3: Centralized architecture

A more complex architecture uses a *Distribution Broker,* shown in figure 4*,* to respond to requests from Context Consumers for context information. The actual information would be stored and provided by Context Sources, but applications can still access all information through the Distribution Broker. To make this work, Context Sources must register the information they can provide with the Context Registry. For example, in a parking application, Context Sources could be parking houses which each can provide context information about numerous entities (parking spaces they enclose, or cars parked within). In addition, Context Consumers can themselves directly discover Context Sources via the Context Registry.

This type of architecture might be selected for a geographically distributed system, or where different partners are responsible for different data repositories. When a Context Consumer requests information from a Distribution Broker, it checks the Context Registry to discover which Context Sources have the appropriate data. For example, a parking application might need to know which parking house contains handicapped parking spaces. The Distribution Broker gathers data from Context Sources and potentially aggregates it for delivery to the requesting consumer. In another mode, all the context information is stored by the Context Sources, and they can directly provide context data to Context Consumers.

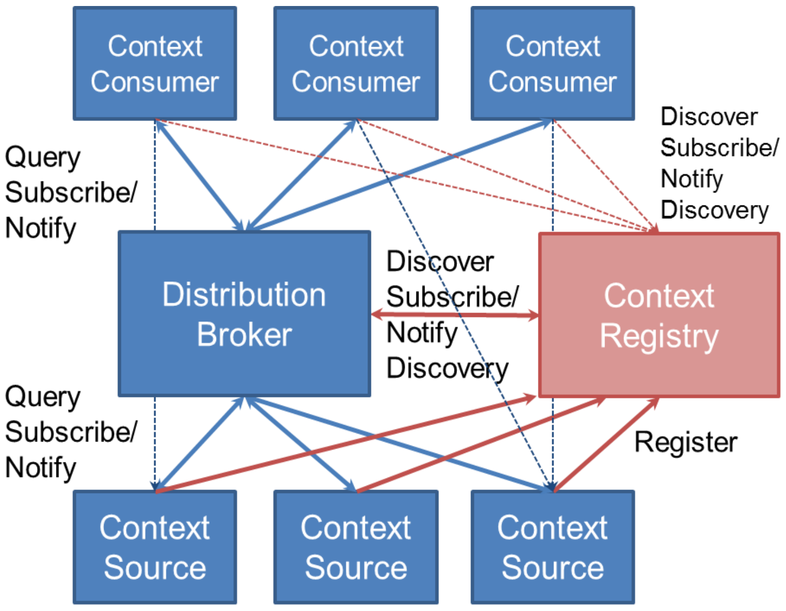

Figure 4 Distributed architecture example

NGSI-LD also supports a Federated Architecture. This is a model for aggregation of NGSI-LD infrastructure in order to extend access to context information across multiple NGSI-LD systems. In the Federated model, NGSI-LD domains are registered to a *Federation Context Registry.* Applications can then query or subscribe to entities within a wider geographic scope. The *Federation Broker* discovers the domain *Context Brokers* that can provide relevant information, forwards the request to these *Brokers* and aggregates the results, so the application gets the result in the same way as in the centralized and distributed cases. Federation can be used for scalability, and also partnering with databases which need to limit access to specific parts of their content (e.g. police data).

Business Model

The NGSI-LD API makes no assumptions about which business models are relevant to the context exchange [9]. The range of deployment architectures can flexibly support various business models and allow them to evolve. The Central Broker architecture is suitable where a single trusted context producer aggregates all the information for delivery to applications, e.g. in a town or municipality, a storm water flow sensor network might be integrated with a rainfall measurement system and an emergency response dispatch system.

The Distributed architecture might be chosen when combining information from a range of integrated public services (e.g. health-care, social care and community services) in a town. The Federated architecture might integrate e.g. traffic control data and parking availability data between neighbouring towns. If the data exchange stops, or some systems are offline, the federate community continues to operate, with one less source of context information.

A Rich Information Model Built Upon Linked-Data Standards.

We now consider the NGSI-LD representation of real-world entities and their context information within a layered information model (see figure 5), consisting of the Core Meta Model, the Cross-domain Ontology, and the open-ended domain-specific models.

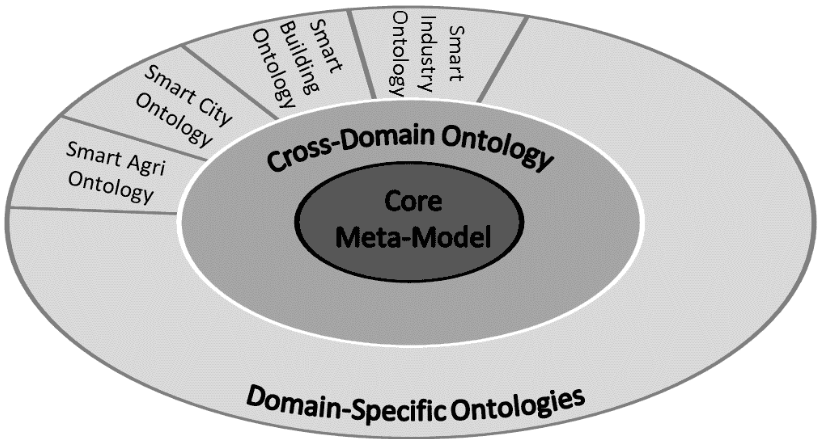

Figure 5 – Layered approach used to define NGSI-LD Information Model

In this model, the NGSI-LD **Core Meta Model**, in the centre of the figure, represents Entities, their Relationships, and their Properties with values. It contains the core terms needed to uniquely represent the key concepts of the NGSI-LD Information model as well as the terms that define the API-related Data Types. These are encoded using JSON-LD, which provides the advantage of being familiar and accessible to developers. For example, an Entity, representing a device or other data source, is encoded using a JSON-LD object.

The **NGSI-LD Cross Domain models** provide commonly used constructs such as time and geographical location. These are generally applicable to many domains so standardising their representation provides valuable cross-domain interoperability. The main features of the NGSI-LD Cross Domain model have been added to distinguish between: (a) the location of an entity; (b) the entity's observation space ("the geographic location that is being observed by the entity", for example in the case of a camera the observation space is different from the actual location of the camera); and (c) the operation space ("the geographic location in which an Entity is active", for example a crane can have a certain operation space).

Proprietary variations of the terms defined in the NGSI-LD Core and Cross Domain models can of course be defined by users, but the NGSI-LD API queries and operations will not directly reference them. For example, even if "yesterday" were to be given a proprietary definition, the query "temperature measured yesterday" would not be valid. The query would need to be expressed with a time interval expressed using the appropriate representation based on ISO8601 (see NGSI-LD specification [1], section 4.6.3). The aim is to keep the Core and Cross-Domain information models as simple and generic as possible. However, ETSI ISG CIM will consider proposals for additional "cross-domain" terms, if they provide clear benefits to the API.

Domain-Specific Ontologies for entities (real world devices, databases, or other information sources) can be created by extending the Cross Domain Ontologies and the Core Meta Model, with specialised terms drawn from other ontologies.

The underpinning of the Meta Model is the Property Graph, which can be formally represented using the RDF model [10]. That core meta-model and cross-domain ontology, illustrated in figures 6a and 6b, make it easy to create real-world models of entities, interrelationships and properties; moreover, the information model is expressive enough to connect and federate other models.

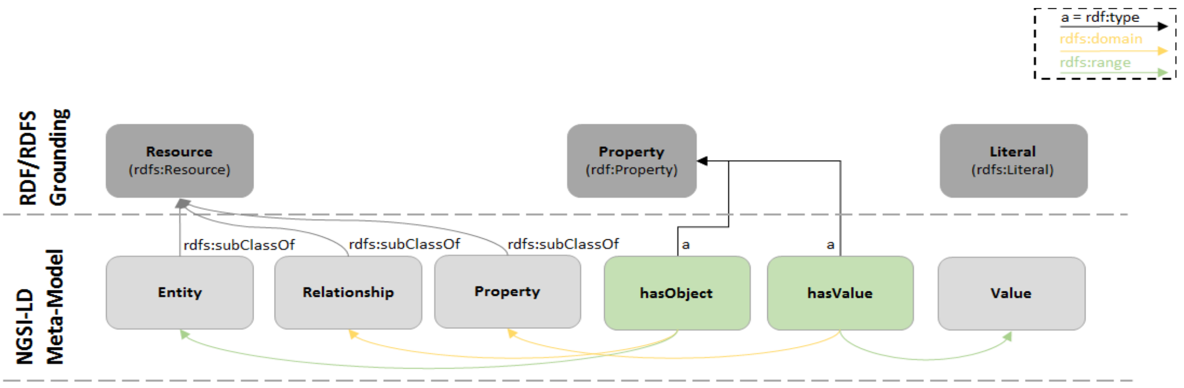

Figure 6a: NGSI-LD Meta-Model

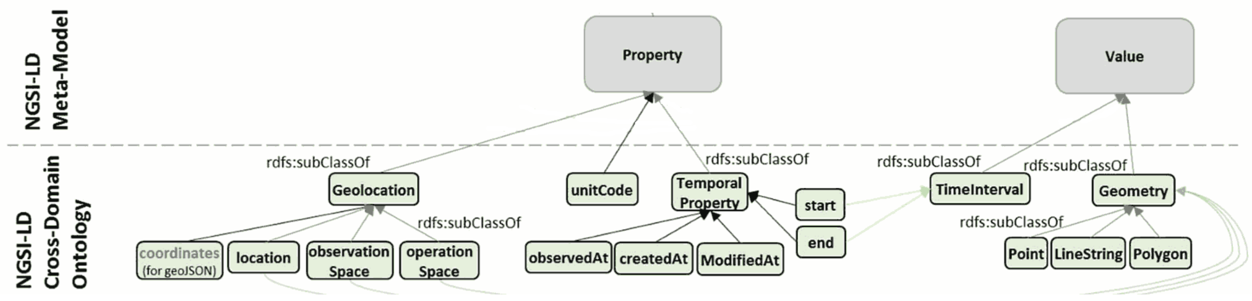

Figure 6b: Cross-domain NGSI-LD Information Model

Property Graph Approach to Model Context Information Use Cases

ETSI ISG CIM uses Property Graphs for the information model to depict the context required to support the example use cases [2]. The property graph enables complex modelling of a relationship with a relationship and also of a property of a property. For example, Ann and Bob can have a relationship “marriedTo” which can have a relationship “witnessedBy” to Charles, or “marriedTo” can have a property “atLocation”; furthermore, the property “atLocation” can have the property “accuracy”. The property graph visualization shows this kind of metadata by the addition of extra arcs between e.g. an existing arc (relationship) and an entity or property. For a wealth of material about Property Graphs, see the websites of any popular graph database such as Neo4j, OrientDB, Titan, BlazeGraph[5]

For each real-life Use Case, the generic Property Graph can be filled in with all the specific details of the particular relationships and properties in that Use Case, to visualize the actual so-called Instance Graph.

Use Case Example - Smart Street Light System

In the use case Smart Street Light System, street lighting levels are controlled by an intelligent agent which reacts to ambient light and weather conditions, traffic levels, and the state of traffic lights in the local area. This application seeks to minimize energy consumption and maintain light levels sufficient for current traffic conditions or pedestrian safety. A simplified Instance Graph is shown in figure 7.

The figure uses some conventions about the shapes enclosing text labels to indicate their function. The four large shaded ovals represent the cooperating domains (e.g., “Street Light Management System”). Entities in each domain (e.g., “Zone A Lights”) are shown in unshaded rectangles. Properties of entities are shown as ovals (e.g., hasState). The values of a property of an Entity are shown in irregular hexagons (e.g., “Brightness = 60” is the value of the property called “hasState” of the entity “Zone A Lights”). Relationships between entities are shown as lines with a name shown inside a diamond shape (e.g. “connected to”).

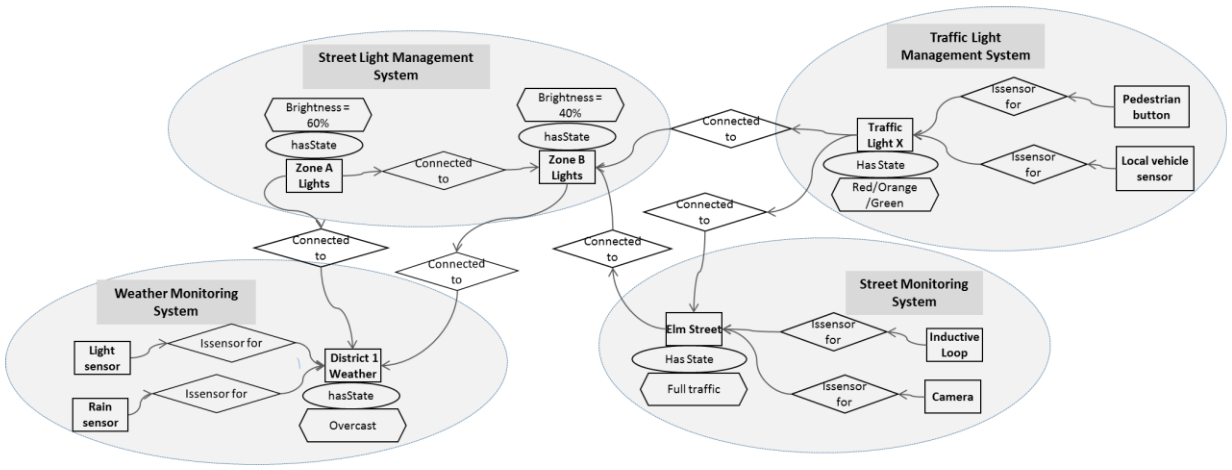

Figure 7: Instance Graph for Smart Street Lighting Application

The example in figure 7 shows four interrelated systems. Entities of the Street Light Management System are related to (have logical or physical proximity to) other Entities within that System, and also to Entities of other Systems, such as a Weather Monitoring System, a Street Monitoring System, and a Traffic Light Management System. Entities (e.g. “Zone A Lights” are represented in the diagram by solid rectangles with incoming and/or outgoing curved lines which indicate that they are subjects and/or objects of relationships with other Entities. Any Properties of one of the Entities are represented by ovals in the figures, e.g. as in

[5] See also https://db-engines.com/en/system/Blazegraph%3BOrientDB%3BTitan for a cross-comparison of features.

"HasState", with the actual values shown inside hexagon shapes. Relationships between subject Entities and object Entities are shown as lines with the name of the relationship, written as a verb such as "contains, isConnectedTo, IsSensorFor", etc, inside a diamond shape. These relationships thus encode logical, physical, or hierarchical links between Entities. Any relationship verb can be defined individually in various domain specific ontologies.

The Weather Monitoring System, the Street Monitoring System, and the Traffic Light Management System each use the NGSI-LD API to make available the information they collect. The Street Light Management System can then query for relevant context information from all of these other systems and may subscribe to notifications and updates.

Other applications and systems, such as the Traffic Light Management System and many other systems might also make use of the context information produced by the various systems and Entities shown in figure 7, as was introduced in figure 2.

NGSI-LD API

Basic Operations

The NGSI-LD API supports a number of operations, with messages expressed using JSON-LD [6]. It allows context consumers and context producers to interact with context information systems. Not all conceivable operations are supported in the API, but rather a subset which is as simple as possible yet complex enough to handle the vast majority of interactions.

For representation in the NGSI-LD API, any Entity is represented by a JSON-LD encoded object. The JSON-LD representation of an Entity includes a reference (in the @context statement) to the NGSI-LD Meta-Model (see figure 6a) along with the specific Entity Type Name, the Entity URI, the Properties and the Relationships associated with that Entity. Each Property includes a Property type, and a value (JSON data type or JSON object). Each Relationship includes a Relationship type, and the object of the Relationship (e.g., another entity). An important characteristic of NGSI-LD is that Properties and Relationships (which are together termed Attributes) may themselves also have Attributes.

Particularly, the API operations allow applications to discover, query and explore the graph-based data by specifying any combination of entities, types, relationships and/or properties as criteria for data queries.

An HTTP REST binding of the NGSI-LD operations is defined in the NGSI-LD API specification [1].

NGSI-API Operations - Context Producers and Consumers.

One group of NGSI-LD operations allow Context Producers to create NGSI-LD Entities i.e. insert an object with a defined URI into the system, and to allow Context Consumers to retrieve and subscribe to Entities. These are as follow:

- Context Information Provision – a set of operations through which a Context Producer can create, modify, and delete an NGSI-LD Entity.
- Context Information Consumption – operations through which a Context Consumer can retrieve or query for NGSI-LD Entities. Queries can filter out Entities by Attribute Values (target value of a Property or the target value of a Relationship).

- Context Information Subscription – operations through which regular or event-driven update notifications of the context of one or more Entities can be created, updated, retrieved, queried for.

Figure 8 shows how these operations can be used in an application. Each horizontal arrow represents the exchange of an NGSI-LD message. A Context Producer has the initial task of inserting/creating an NGSI Entity in the NGSI-LD System by transferring a JSON-LD message representing Entity A to the system. A Context Consumer later queries the NGSI-LD System for Entities meeting particular conditions. The Context Consumer is informed that Entity A meets the conditions, and then subscribes to notifications from Entity A. When information about Entity A is next updated by the Context Producer, the System automatically informs the Context Subscriber of the new value.

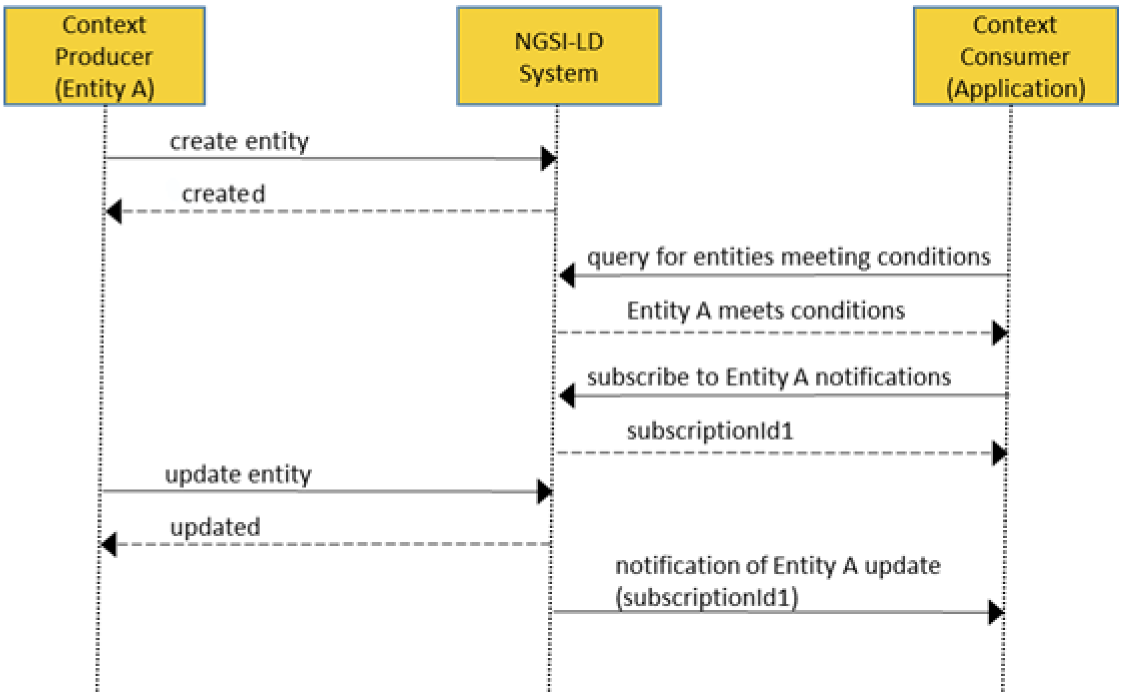
Context Producer (Entity A)
NGSI-LD System
Context Consumer (Application)
create entity
created
query for entities meeting conditions
Entity A meets conditions
subscribe to Entity A notifications
subscriptionId1
update entity
updated
notification of Entity A update (subscriptionId1)

Figure 8 – Example API Operations (Context Producer and Context Consumer)

NGSI-API Operations - Context Sources and Consumers.

Another group of NGSI-LD operations allow Context Sources to be registered as potential sources of information meeting certain conditions. A Distribution Broker can query a Registry to ascertain which Context Sources may be able to provide the information requested.

- Context Source Registration – a set of operations through which a Context Source (i.e., the entire collection of information which it could provide) can be registered, updated, and deleted (removed from the registry). The registration information includes the types of Entities, Properties, and Relationships about which the Context Source can provide information, as well as geographic and temporal constraints on the information (e.g., “only in the region Germany”, “only for years 2017 and later”). For example, a particular Context Source could register that it can provide the indoor temperature for Building A and Building B or that it can provide the speed of cars in a geographic region covering the centre of a particular city.

- Context Source Discovery – operations through which a Context Consumer or Producer can retrieve or query Context Source registrations.

- Context Source Registration Subscription – a set of operations through which a Context Consumer can create, update, retrieve, query for, or be notified regarding Context Source registration subscriptions. In other words, subscribers may be notified about new Context Source Registrations that can potentially provide the requested information.

Figure 9 shows the sequence of NGSI-LD message exchanges which would allow a Context Source for smart street lighting to: (a) register its availability to a Broker; (b) for a Context Consumer to discover the existence of a Context Source (in the right town, with up-to-date data); then (c) query for updates of the status of a specific entity (e.g. the “red/green” status of the traffic signals along a particular street). Such a query would allow, e.g., an ambulance to query the red/green status of traffic signals on the route ahead. Alternatively, a city planner might add some simple counting software to check the daily average intervals when pedestrians have right-of-way across the street.

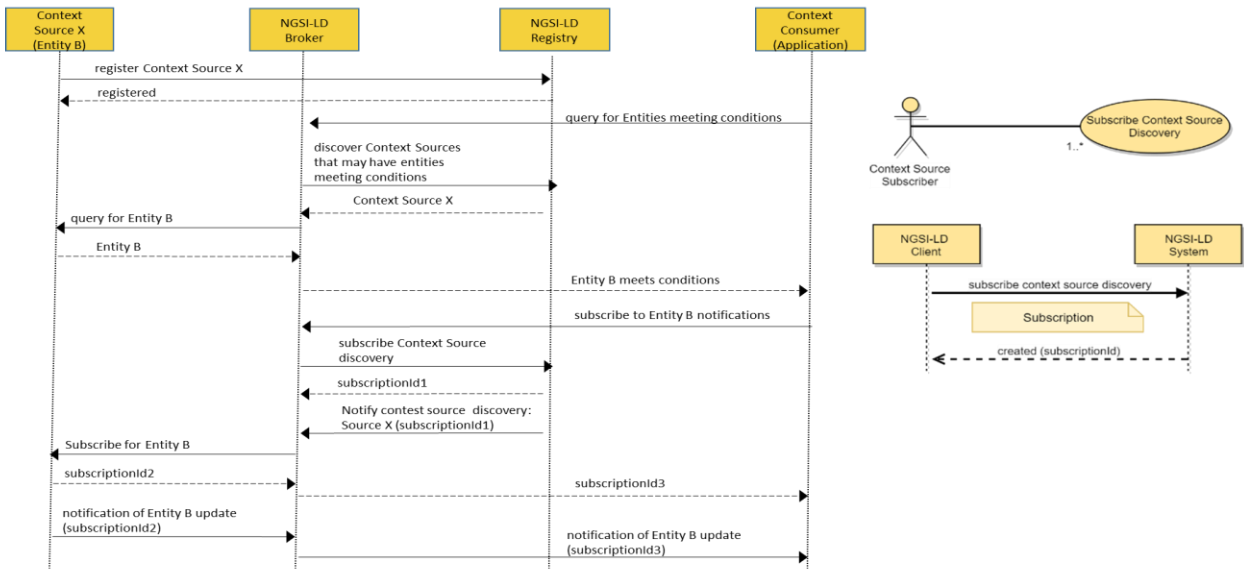
Context Source X (Entity B)
NGSI-LD Broker
NGSI-LD Registry
Context Consumer (Application)
register Context Source X
registered
query for Entities meeting conditions
discover Context Sources that may have entities meeting conditions
Context Source X
query for Entity B
Entity B
Entity B meets conditions
subscribe to Entity B notifications
subscribe Context Source discovery
subscriptionId1
Notify contest source discovery: Source X (subscriptionId1)
Subscribe for Entity B
subscriptionId2
subscriptionId3
notification of Entity B update (subscriptionId2)
notification of Entity B update (subscriptionId3)
Context Source Subscriber
Subscribe Context Source Discovery
1..*
NGSI-LD Client
NGSI-LD System
subscribe context source discovery
Subscription
created (subscriptionId)

Figure 9 – Sample API Operations (Context Source and Context Consumer)

The NGSI-LD API and its Information Model (Core Meta-Model and Cross Domain Meta-Model) provide a powerful framework, but they do not provide detailed vocabulary, definitions and models for every real-life application. NGSI-LD does allow – indeed it mandates – that every term is uniquely defined, by reference to a specific vocabulary, avoiding the possibility of confusion of naming. Here the advantage of using JSON-LD becomes very apparent, because the terms can be defined in a separate document, referenced by an @context statement. The external resource document can be as simple or as complex as needed in an application or a domain.

Summary of advantages of NGSI-LD

Although various data exchange methods could be selected for smart services, the following points are substantial advantages of the NGSI-LD approach for solution providers:

- Cities wanting to expose their open data, for use of citizens or even disparate city departments, will find NGSI-LD useful for publishing (parts of) existing files and PDF documents using explicit and exchangeable vocabulary and outputting parts of existing “data-lakes”

- Vendor lock-in can be avoided as the NGSI-LD framework can be easily extended to accommodate specific features of each application in the domains of e.g. Smart City, Smart Industry or Smart Agriculture.
- The API is independent of choice of architecture so that multiple independent systems can easily link up, and partners with data can join or leave without disruption to services.

From the viewpoint of software developers, the following advantages are important:

1. A text-string based serialization of data and its context, using developer-friendly JSON-LD.
2. An explicit procedure for relating every term used in context information exchanged between systems to an unambiguous definition (using the @context statement of JSON-LD).
3. A very flexible high-level information model which can directly represent data from a wide variety of information sources.
4. A simple set of operations via the API which can be used across dynamic, federated databases to register and discover data sources based on graph patterns. The data sources can include existing "data lakes" which may have been deployed e.g. for cities.
5. A query language, independent of the underlying database technology(s), to unambiguously detect, filter, and set up subscriptions for updates to selected data.

Ongoing Work and Getting Involved

The first phase of the ETSI ISG CIM specification work was completed mid-2018. The initial cornerstone documents describing the NGSI-LD API [1] as well as an accompanying Use Case Analysis [2] are published.

ETSI ISG CIM welcomes participation in this specification development work by the IoT community and smart city stakeholders, as well as those in other domains such as smart agriculture, smart industry, service providers, vendors of related infrastructure, and software developers.

As the work continues, the next steps for ISG CIM are:

- To provide new or revised standards documents, with a target of year-end 2019:
  - NGSI-LD API functionality may be extended in key areas such as query capabilities.
  - An Information Model Document will be published, specifying further details of the Property Graph approach, guidelines for using it in data models, and cross-references to useful models from a number of different domains.
  - An analysis of data security and privacy requirements for NGSI-LD will be published.
- To encourage implementations and enhance tie-ins with related industrial and enterprise projects focussing upon context information [9] to drive a new generation of services:
  - The FIWARE Foundation open-source project will contain an adaption module for interaction with NGSI-LD API and is committed to natively supporting NGSI-LD.
  - The Orange 'Thing' in project [11] will make use of the NGSI-LD information model to achieve an object-oriented vision of smart services.
  - Collaborations with a number of EC Horizon2020 projects will provide feedback for improving the NGSI-LD API.

ETSI ISG CIM heartily welcomes input, critique and collaboration and looks forward to your contact.

Relations to ETSI TC SmartM2M and oneM2M

ETSI ISG CIM makes no attempt to handle sensors and M2M device management, device connectivity, gateway management, interworking of different M2M devices, etc. The focus of ISG CIM is to enable exchange of information and metadata and "to develop for cross-domain context information management an interchange protocol between diverse systems from the IoT, Mobile Applications, Open Data Portal and proprietary database worlds". The work in GS CIM 009 NGSI-LD API [1] is to combine Linked Data and Open Data principles using an API based on an information model using Property Graph concepts to combine information from user Apps, municipal databases, open data portals and IoT platforms (including oneM2M).

NGSI-LD is expanding potential usage of oneM2M and supporting use of ETSI SmartM2M (SAREF) work to align ontologies across many domains. Discussions are underway on how to achieve aspects of NGSI-LD functionality via the oneM2M interfaces.

In its terms of reference, ETSI ISG CIM includes:

- report considering how NGSI-LD functionality could be proposed into oneM2M (mechanism could be e.g. liaisons, workshops or direct contributions by Members who are in both organisations) and might involve collaboration on new or existing Work Items
- consolidate and encourage re-use of a cross-domain ontology (collaborating with SAREF in cooperation with SmartM2M, or also e.g. schema.org, or W3C, etc.) which will provide for re-use of identical concepts and preclude misinterpretation of exchanged information
- collaborate with oneM2M concerning means to interwork with oneM2M systems today and consider feasibility of including NGSI-LD functionality via the Mca Reference Point in the future

Acknowledgements

The authors thank the following people who were co-creators of the ISG CIM information model and NGSI-LD API discussed in this whitepaper, as well as their companies and the EU research projects which supported their work: Abdullah Abbas (Orange), José Manuel Cantera (Fiware Foundation), Gilles Privat (Orange).

Related European Union Horizon 2020 research and innovation programmes are: No. 732851 (FI-NEXT), No. 723156 (WISE-IoT), No. 732240 (SynchroniCity) and No. 731993 (AutoPilot),

References

[1] ETSI GS CIM 009 V1.1.1 (2019-01) Published. Context Information Management (CIM); Application Programming Interface (API). See http://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM004v010101p.pdf

[2] ETSI GR CIM 002 V1.1.1 Context Information Management (CIM); Use Cases (UC). See https://www.etsi.org/deliver/etsi_gr/CIM/001_099/002/01.01.01_60/gr_CIM002v010101p.pdf

[3] Open Mobile Alliance, Next Generation Service Interfaces Architecture Approved Version 1.0 Published 29 May 2012

[4] Fiware, FIWARE-NGSI v2 Specification. See e.g. http://fiware.github.io/specifications/ngsiv2/stable/

[5] oneM2M, TS-0001-Functional_Architecture-V2.18.1 See http://www.onem2m.org/images/files/deliverables/Release2A/TS-0001-Functional_Architecture-v_2_18_1.pdf

[6] W3C, JSON-LD 1.1 A JSON-based Serialization for Linked Data. See https://www.w3.org/TR/json-ld11/

[7] Rodriguez, M.A., Neubauer, P. Constructions from dots and lines. Bul. Am. Soc. Info. Sci. Tech. 36(6), 35{41 (2010). See https://pdfs.semanticscholar.org/2b21/1f9553ec78ff17fa3ebe16c0a036ef33c54b.pdf

[8] Rodriguez, M.A., Neubauer, P. The graph traversal pattern. Tech. rep., AT&T and NeoTechnology (April 2010). See https://pdfs.semanticscholar.org/ae6d/dcba8c848dd0a30a30c5a895cbb491c9e445.pdf

[9] Haller S., Karnouskos S., Schroth C. (2009) The Internet of Things in an Enterprise Context. In: Domingue J., Fensel D., Traverso P. (eds) Future Internet – FIS 2008. FIS 2008. Lecture Notes in Computer Science, vol 5468. Springer, Berlin, Heidelberg. See https://www.alexandria.unisg.ch/46642/1/fis2008-haller-final.pdf

[10] W3C, Resource Description Framework (RDF), consists of a suite of documents. See https://www.w3.org/standards/techs/rdf#w3c_all

[11] Orange, Thing'n http://www.thinginthefuture.com/

ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org