Journal of Internet Services and Applications

RESEARCH Open Access

Interoperability between OPC UA and oneM2M

Salvatore Cavalieri* and Salvatore Mulè

* Correspondence: salvatore.cavalieri@unict.it
Department of Electrical Electronic and Computer Engineering, University of Catania, 95125 Catania, Italy

Abstract

A key requirement of realizing the connected world featured by IoT is to ensure interoperability among different connected devices. Interoperability is also at the basis of the realization of the novel vision of Industry 4.0; a lot effort is put to make interoperable the interchange of information between industrial applications, also including IoT ecosystems. For this reason, during these last years, several approaches aimed to enhance interoperability between industrial applications and IoT appeared in the literature. In this paper an interoperability proposal is presented. It is based on the idea to realize interworking between the two standards considered among the reference ones in the industrial and IoT domains. They are the OPC UA for the industrial domain and oneM2M for the IoT. Interworking is realized in such a way to allow industrial applications based on OPC UA to acquire information coming from oneM2M-based IoT devices. The proposal allows an OPC UA Server to publish each piece of information produced by oneM2M-based IoT devices, so that this information may be consumed by industrial applications playing the OPC UA Client role.

Keywords: Interoperability, Interworking, OPC UA, oneM2M, Industry 4.0, IoT

1 Introduction

During the last two decades, web technologies played a very important role for the definition of novel software solutions in the web. Web technologies have shifted the web from pure information supply to a social platform and ushered in Web 2.0 to enable seamless information integration, crowd intelligence, and nearly effortless content creation without the need for specialized technical skills or devices. The current phase of the web evolution features Machine-to-Machine (M2M) and Internet of Things (IoT) technologies [1, 2].

M2M is a point-to-point connection between two network devices that allows them to transmit information via public networking technologies such as Ethernet and cellular networks; sensor telemetry is one of the original uses of M2M communication. IoT is an evolution of M2M; it takes the basic concepts of M2M and expands them outward by creating large "cloud" networks of devices that communicate with one another. Examples of IoT devices are all around us today: any network of devices that is connected to the Internet and uses a cloud platform to communicate can be

considered part of the IoT. The biggest difference between M2M and IoT is that an M2M system uses point-to-point communication. An IoT system, meanwhile, typically situates its devices within a global cloud network that allows larger-scale integration and more sophisticated applications. Scalability is another key difference between M2M and IoT. IoT is designed to be highly scalable since devices can often be added to a network and integrated into existing systems with minimal hassle.

IoT has crept into our everyday life. One of the most evident examples refers to the home automation scenario, featured by a lot of appliances such as WiFi-enabled programmable thermostats or LED light bulbs embedded with a ZigBee radio chip; such home appliances are directly, or via a gateway, connected to the Internet so that end-users can access them remotely from anywhere at any time. The advances in IoT are changing our life aiming to make it more "smart"; considering again the home automation scenario, home appliances and utilities connected each other can share the information about the changes in their status and surroundings, and thus provide home occupants with smart services in a proactive and intelligent manner [3].

Also in the industrial scenario (e.g. manufacturing sector) IoT is considered a key point to realize the "smart" factory where all things in the factory itself as well as throughout the supply chain are fully connected and thus almost real-time data analytics and insights could be generated, helping manufacturers adapt their facilities and assets accordingly, maintain workforce productivity efficiently, and manage their supply chain proactively [4]. Since few years, industry has been involved in a revolution, the fourth one, usually known as Industry 4.0. It features the application of modern Information and Communication Technology concepts, mainly those relevant to the IoT, in industrial contexts to create more flexible and innovative products and services leading to new business and added value models [5, 6].

Interoperability among various devices including smart devices and resource constrained devices is imperative requirements for IoT realization. However, it seems a challenging task to make different types of devices interoperable because they will have a wide variety of heterogeneous hardware and software systems. For this reason, several leading standard development organizations have been working on developing standards for solving the fragmentation of IoT landscape; among the novel standards for IoT there is that called oneM2M [7]. Several pieces of oneM2M-based platform such as Eclipse OM2M [8], nCube [9], Mobius [10–12] and Secure OM2M Service Platform [13] were developed and realized, in most case as open source solutions. Many of the available oneM2M standard-based IoT platforms have shown its practical feasibility in developing home appliances [9], smart farm [14], smart office [15], medication [16], and semantic interworking in smart cities [17, 18]. Literature presents several papers presenting proposals for improving performances of oneM2M-based platforms [19, 20].

Interoperability is also at the basis of the realization of the novel vision of Industry 4.0; a lot effort is put to make interoperable the interchange of information between industrial applications [21]. As the definition and adoption of communication standards are of paramount importance for the realization of interoperability, during the last few years, different organizations have developed reference architectures to align standards in the context of the fourth industrial revolution. One of the main examples is the "Reference Architecture Model for Industry 4.0 (RAMI 4.0)" [22]. Another one is the Industrial Internet Reference Architecture (IIRA) by the Industrial Internet Consortium (IIC)

[23, 24]. Both architectures consider OPC UA (Open Platform Communications Unified Architecture) [25] as one of the main reference communication standards for the industrial applications.

As said before, among the main goals of the Industry 4.0 there is the interoperability between industrial applications and IoT ecosystems. This interoperability may be enhanced by interworking solutions between communication standards in industry and IoT domains. As the current state-of-the-art just presented pointed out that OPC UA and oneM2M are leading communication standards in these two domains, authors believe that interworking solution between OPC UA and oneM2M could be a valid solution to enhance integration of industry applications and IoT systems. In the industry domain, applications based on OPC UA commonly play the client role accessing data maintained by one or more OPC UA Servers. One of the possible ways to allow these applications to acquire information coming from oneM2M-based IoT devices is that to enable interworking from oneM2M towards OPC UA. What is required is that information produced by oneM2M-based IoT devices could be published by an OPC UA Server and so consumed by industrial applications playing the client role. According to this scenario, industrial applications may acquire information coming both from traditional industrial sensors/devices and IoT devices.

On account of what written, the paper proposes a novel solution to realize the interworking between OPC UA and oneM2M, in the direction from oneM2M to OPC UA. The interworking solution is based on a mapping of each information produced in the oneM2M ecosystem into the OPC UA domain.

The paper is organized as follows. Section 2 points out the related works about interoperability between industrial applications and IoT ecosystems. Section 3 gives an overview on OPC UA and oneM2M protocols. Section 4 introduces the interworking solution proposed by the authors. Section 5 points out the main details of this proposal. Section 6 presents a case study in order to better understand the proposed solution. A final remark section gives the author's conclusions and describes the software implementation of the proposal made by the authors.

2 Related work

As interoperability of industrial applications and IoT ecosystems is one of the main goals of the Industry 4.0, several approaches about this issue appeared in the literature in these last years. Due to important role played by OPC UA inside the current Industry 4.0 reference models, many approaches deal with the interoperability of OPC UA and IoT ecosystems. The aim of this section is that to give an overview of the state-of-the-art about interoperability between industrial applications and IoT domain, focusing on solutions based on OPC UA.

A common theme present in literature is vertical integration, where solutions for achieving interoperability involves OPC UA and Device Profile for Web Services (DPWS) [26–28]. Among them, the solution proposed for enabling interoperability with OPC UA described in [28] uses a gateway approach bridging DPWS and OPC UA networks.

Arguments for direct modification of OPC UA to better adapt it to the industrial IoT have also been raised in [29, 30]. Specifically, the research activities reported by these papers involve a series of adaptations to the OPC UA binary protocol, enabling stateless

service requests and reducing communication overhead thereby making it more RESTful, and more friendly to resource-constrained devices.

Similarly, CoAP has been proposed as a transport option for the OPC UA stack [31]. In [32] a proposal for an OPC UA translator between OPC UA and other communication systems used in IoT (i.e. HTTP, CoAP and MQTT) has been presented.

Technical report [33] introduces a very preliminary work about the interoperability between OPC UA and oneM2M standard; the paper [34] presents additional results on the same subject. It is important to point out that both [33, 34] only deals with the interoperability from OPC UA towards oneM2M; exchange of information in the opposite direction is not considered at all. As pointed out in the introduction, the aim of the paper is to define a solution in charge of enabling interworking from oneM2M towards OPC UA, allowing that information produced by oneM2M-based IoT devices could be published by an OPC UA Server and so consumed by industrial applications playing the client role. This solution cannot be realized with the interoperability proposal presented in [33, 34].

The authors have published very preliminary results about the proposal here presented [35, 36]; the results given in these papers were relevant to a very early stage of the research; more important these results were not fully validated. This paper will present the final results of the research carried out by the authors; these results are now supported by a real software implementation of the proposal which will be presented in the paper. This implementation allowed a full validation of the outcomes presented in the paper.

To the best of author's knowledge, no other papers are present in the current literature dealing with the interworking between OPC UA and oneM2M. In order to better highlight this last concept, Table 1 summarizes the current state-of-the-art described before, pointing out the main existing approaches aimed to make OPC UA interoperable with IoT domain.

3 Overview the OPC UA and oneM2M protocols

The aim of this section is that to describe the main features of the OPC UA and oneM2M protocols. The description will be limited to the properties which are mainly involved in the interoperability solution here presented: Information Model and data access - oriented services. For each of these properties, the features of the two protocols will be compared, into separate subsections.

Table 1 Interoperability solutions involving OPC UA and IoT domain

Protocol	Description of the interoperability solution	Reference
DPWS	Several solutions for achieving interoperability between OPC UA and DPWS exist	[26] [27] [28]
REST	Solutions aimed to modify OPC UA binary protocol, making it more RESTful, and more friendly to resource-constrained devices.	[29, 30]
CoAP	Solution to make interoperable OPC UA and CoAP exixts	[31, 32]
HTTP	A proposal for an OPC UA translator between OPC UA and HTTP exists	[32]
MQTT	A proposal for an OPC UA translator between OPC UA and MQTT exists	[32]
oneM2M	Interoperability approaches between OPC UA and oneM2M standard exist, but they only deal with the interoperability from OPC UA towards oneM2M; exchange of information in the opposite direction is not considered at all.	[33, 34]

3.1 Information model

The main purpose of an Information Model is to model managed objects at a conceptual level, independently of any specific implementations or protocols used to transport the data [37]. Understanding OPC UA and oneM2M Information Models is very important as the interoperability solution here presents is based on the mapping of information model objects from oneM2M domain towards OPC UA one.

3.1.1 OPC UA

OPC UA is an international standard (IEC 62541), mainly based on a client/server communication model allowing distribution to OPC UA Clients of information maintained by an OPC UA Server [25]. The set of information is organized through OPC UA Nodes grouped together to compose the so-called AddressSpace inside an OPC UA Server. Each OPC UA Node belongs to a class named NodeClass [38].

Among the available NodeClasses, there is the Variable NodeClass allowing to maintain a value, by an attribute named Value. Variables may be Properties (containing metadata) and DataVariables (containing real values of the system, e.g. measurements coming from sensors). Another NodeClass is the Object modelling real-world entities like hardware and software components of a system, or even a whole system. An OPC UA Object is a container of other OPC UA Objects and Variables; for instance, an OPC UA Variable Node may be a component of an Object in order to represent real values or properties of the Object.

OPC UA defines particular NodeClasses defining types. Among them, there is the DataType which specifies the type of the Value attribute of a Variable. DataType may be for example Built-in or Enumeration. Built-in provides base types like integer (e.g., UInt32 and Int32). Enumeration represents a discrete set of named values.

Another type is the ObjectType NodeClass which holds type definition for OPC UA Objects. OPC UA defines the BaseObjectType which all the ObjectTypes must be extended from. Among standard ObjectTypes derived from BaseObjectType, there is the FolderType whose instances are used to organize the AddressSpace into a hierarchy of OPC UA Nodes. VariableType is another NodeClass used to provide type definition for Variables. OPC UA defines the BaseVariableType which all the VariableTypes must be extended from. Among the standard VariableTypes derived from BaseVariableType, there are the BaseDataVariableType and the PropertyType. The former is used to define a DataVariable Node, whilst the latter defines a Property Node.

Particular relationships may be defined between OPC UA Nodes; they are called OPC UA References. A Reference connects a source Node to a target Node. The ReferenceType NodeClass is used to define the exact type of each Reference. Among the available types, the following ones will be used in the paper. The HasComponent Reference allows to specify that an OPC UA Object contains another OPC UA Object or OPC UA DataVariable. Organizes Reference allows to organize OPC UA Nodes inside a folder made up by a FolderType Object. The HasProperty Reference is used to link a source OPC UA Node to a target OPC UA Property; the meaning is that the source Node features a property described by the target Node. HasSubtype expresses a subtype relationship between types.

A very important role is played by the HasModellingRule Reference. For each OPC UA type, the relevant instances may have some mandatory elements (e.g. a particular Object as component), whilst other elements may be optional (e.g. a certain Property). HasModellingRule Reference allows to give this information for each OPC UA type defined inside the AddressSpace. Definition of an OPC UA type is realized specifying the set of Variables and/or Objects (henceforward called InstanceDeclarations) which a generic instance may potentially hold. For each InstanceDeclaration, a HasModellingRule Reference points to a ModellingRule Object as target Node. The ModellingRule Object specifies whether the relevant InstanceDeclaration must be present or not in every instance of an OPC UA type Node. A Mandatory ModellingRule specifies that instances of the OPC UA type must have that InstanceDeclaration. An Optional ModellingRule Optional, instead, specifies that instances of the OPC UA type may have that InstanceDeclaration, but it is not mandatory. The MandatoryPlaceholder and OptionalPlaceholder ModellingRule Objects also exist; the difference with the previous ModellingRule Objects is that the counterparts of InstanceDeclaration in each instance may be more than one.

OPC UA defines standard graphical representation for both Nodes and References; some of them are summarized by Figs. 1 and 2.

3.1.2 oneM2M

The oneM2M communication system provides interoperability support for IoT technology [7, 39, 40]. According to the oneM2M reference architecture model, the IoT environment are divided into two domains: infrastructure and field. The first is the domain in which servers and applications (e.g. control, monitoring) reside. Field domain contains the oneM2M-compliant IoT devices exchanging data with the servers and applications located at the infrastructure domain; communication with the infrastructure domain may be realized also through one or more gateways located in the Field domain.

Nodes are logical entities identifiable in oneM2M System, which reside in each of the two domains. Typically, a Node contains one or more of the following entities: Application Entity (AE), Common Service Entity (CSE) and Network Services Entity (NSE)

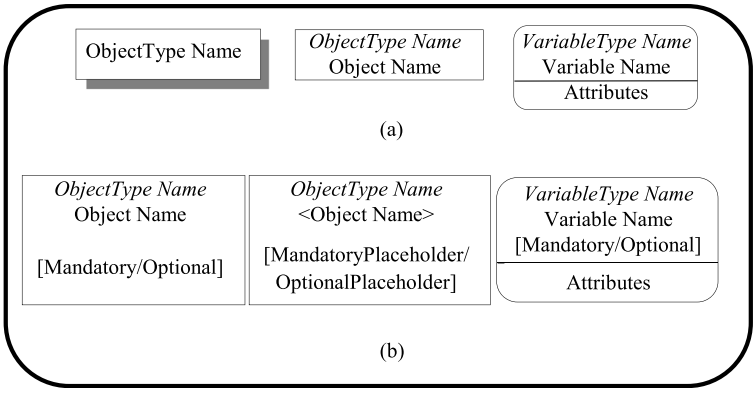

Fig. 1 Standard graphical representation of Nodes in OPC UA: (**a**) ObjectType, Object, DataType and Variable; (**b**) Objects and Variables featuring ModellingRule Objects

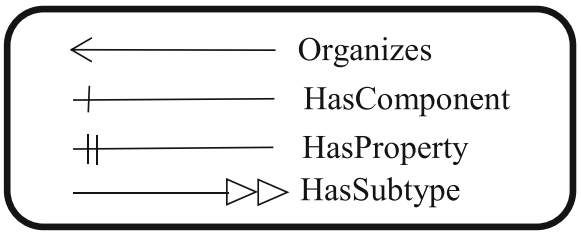

Fig. 2 Standard graphical representation of References in OPC UA

[39]. Application Entity represents application services located in a IoT device, gateway or server. Common Service Entity represents an instantiation of a set of functions with which the oneM2M platform provides common services for the IoT environments. Network Services Entity provides communication services from the underlying network to be utilized by the CSEs. Communication flow between these entities is supported by the so-called reference points. Mca enables communication between AE and CSE. Mcc enables communication between CSEs. Mcn has been defined for the communication flow between a CSE and the NSE; it allows a CSE to use the supported services provided by the NSE.

In the following, an overview on the main types of nodes defined in oneM2M will be given.

Application Dedicated Node (ADN) is a node that contains at least one AE and does not contain a CSE. An ADN would typically be implemented on a resource constrained device (e.g. IoT device). AE contained in an ADN is named ADN-AE.

Middle Node (MN) is a node that contains one CSE (named MN-CSE) and could contain AEs (i.e. MN-AE). Typically, a MN would reside in a gateway.

Infrastructure Node (IN) is a node that contains one AE (i.e. IN-AE) and could contain CSEs (i.e. IN-CSE). An IoT Server is an Infrastructure Node containing both IN-AE and IN-CSE; an application (e.g. running in a smartphone) is typically an Infrastructure Node containing only an IN-AE.

Non-oneM2M Node (NoDN) is a node that does not contain oneM2M entities. Such nodes represent devices attached to the oneM2M system for interworking purpose.

Figure 3 shows the domains defined in oneM2M and the Nodes described above, belonging to each oneM2M domain; the reference points Mca and Mcc between the several entities are also clearly pointed out in the same figure.

The oneM2M architecture adopts the Resource-Oriented Architecture (ROA) model, and thus the services and data that oneM2M system supports are managed and exposed as a resource information model [39, 40]. According to the ROA concept, resources can be uniquely addressed by the Uniform Resource Identifier (URI).

Many resource types are defined in the oneM2M communication system; each of them is made up by set of mandatory and optional attributes [39]. Among the mandatory attributes (called “universal attributes”) there is the resourceType attribute that identifies the type of resource. Another attribute is resourceID that is the identifier of resource; resourceName is the name of resource used to represent parent-child relationship. Attribute parentID identifies the parent resource; creationTime is the timestamp of resource creation. Attribute lastModifiedTime is the timestamp of last modification of the resource.

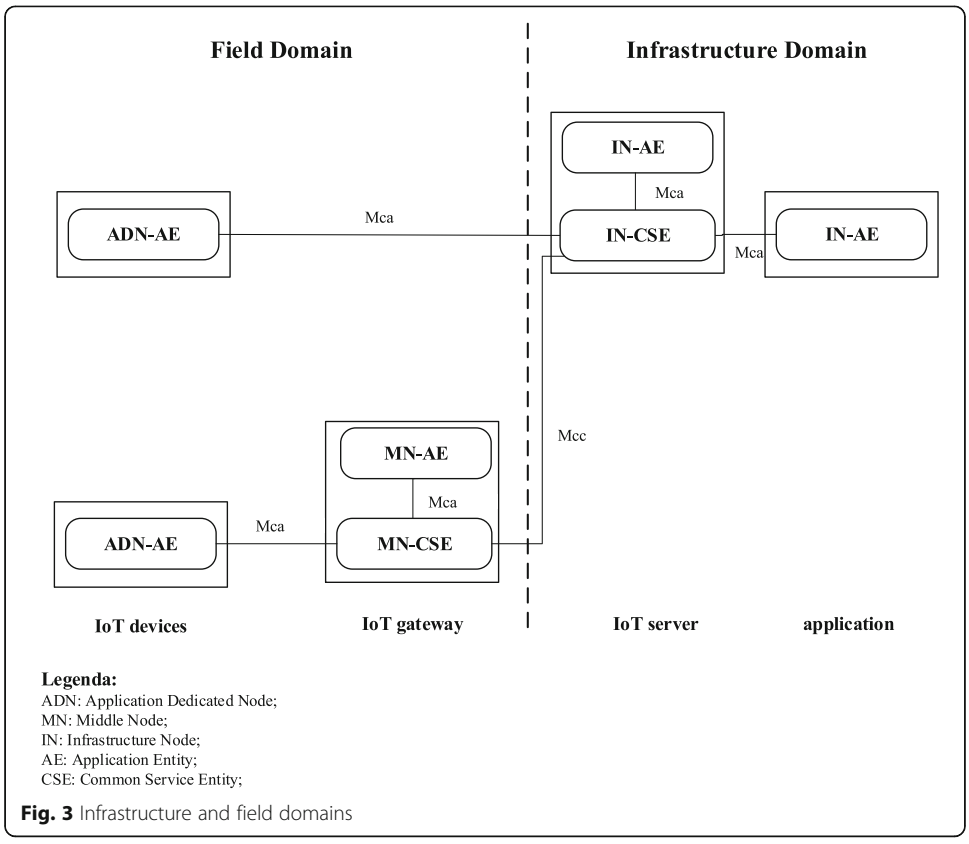

Fig. 3 Infrastructure and field domains

The oneM2M system manages its resources through a hierarchical structure. Resources are created as child of other resources. For this reason, each resource features a hierarchical structure made up by attributes and child resources. Figure 4 shows an example of the hierarchical structure of a generic resource; among the attributes shown by the figure, there are those described before.

Furthermore, oneM2M resources are organized in a resource tree whose root is realized by a particular resource of type <CSEBase>. Figure 5 shows an example of the hierarchical structure of resources inside oneM2M system.

In the following, the resource types used in the paper will be described in more details.

A < CSEBase> resource represents a CSE. A resource of <CSEBase> type is the root of the resource tree which organizes the resources of a oneM2M system.

In order to expose the oneM2M resources, it is necessary that an AE or another CSE must be registered in the CSE hosting the oneM2M resource tree. This is realized by the creation of a < AE> or < remoteCSE> resource, respectively, inside the CSE resource tree. According to [39] a Registrar CSE is the CSE where an AE or another CSE has been registered.

A < remoteCSE> resource represents a CSE that is registered to another CSE. <remoteCSE> resources shall be located directly under the <CSEBase> resource of the CSE where it is registered. An <AE> resource represents information about an Application Entity registered to a CSE.

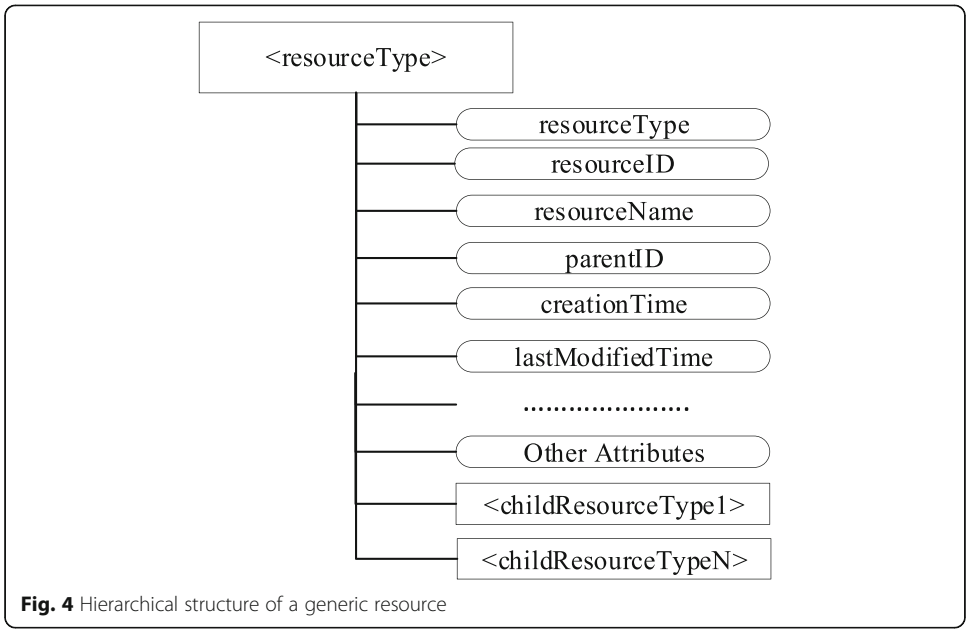

Fig. 4 Hierarchical structure of a generic resource

The <container> resource represents a container for data instances; it is used to share information with other entities and potentially to track the data. The <contentInstance> resource represents a data instance in the <container> resource.

The <group> resource represents a group of resources of the same or mixed types. The <group> resource can be used to do bulk manipulations on the resources represented by the memberIDs attribute.

The <accessControlPolicy> resource (ACP) provides for authorization information. A generic resource which requires the application of a particular access control policy, may link to the particular ACP specifying this policy.

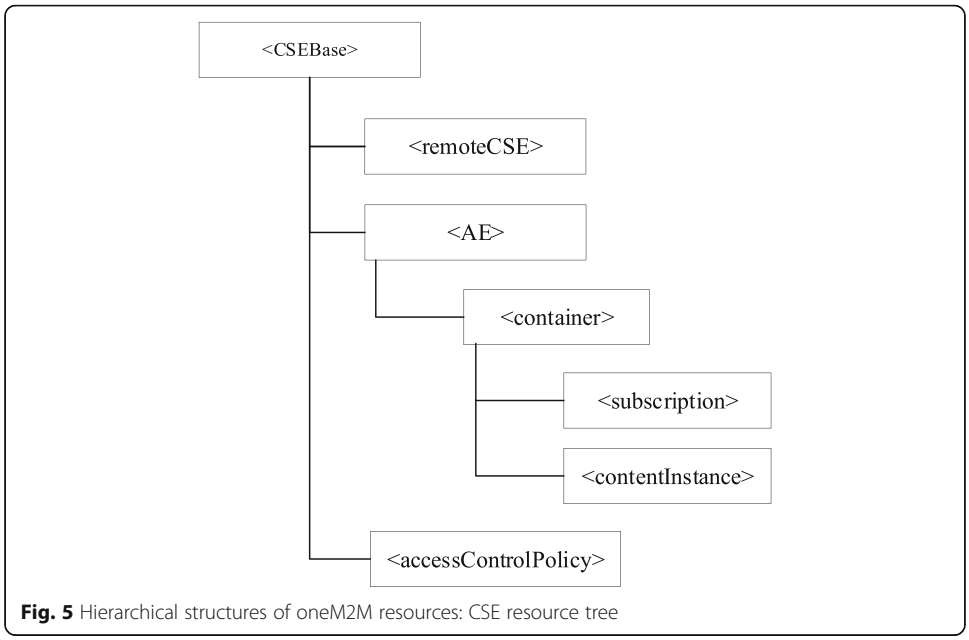

Fig. 5 Hierarchical structures of oneM2M resources: CSE resource tree

The concept of subscription to resource instances in order to receive notifications about content changes is also specified in oneM2M; it allows efficient monitoring of resource instances and thus of the exposed resources. In particular, the resource defined as <subscription> contains subscription information to a particular resource.

3.2 Data access – oriented services

This subsection aims to describe the services available in the two protocols for the data access. Understanding of these services is important as the interoperability approach here presented maps OPC UA services into oneM2M ones and vice versa.

3.2.1 OPC UA

OPC UA offers many services to allow an OPC UA Client to access the AddressSpace of an OPC UA Server [41]. The simplest access which an OPC UA Client may realize is that to browse the AddressSpace of an OPC UA Server by using the OPC UA Browse Service.

The OPC UA Read service is used to read one or more attributes of Nodes. OPC UA Client invoking the OPC UA Read Request may specify a maxAge parameter (expressed in milliseconds). Briefly, the maxAge parameter is used to force the OPC UA Server to access the requested value directly from the underlying data source if the "age" of the current value maintained in the AddressSpace is greater than the maxAge. The age of the value is based on the difference between the ServerTimestamp (i.e. the time at which the local data has been stored in the local AddressSpace) and the time when the Server starts processing the request [41]. More details about the procedures performed by OPC UA Server to handle maxAge parameter will be given in Section 5.3.

The Write service allows the writing of one or more attributes of Nodes. The values are generally written to the data source; the OPC UA Server will report if it succeeds in the write operation. Depending on the particular implementation, the OPC UA Server may write to an intermediate system and the data source will be updated by using other mechanisms external to the standard. In these cases, the OPC UA Server should report a success code that indicates that the writing operation on the data source was not verified.

Subscriptions and MonitoredItems represent a more sophisticated way to exchange data between OPC UA Client and Server. They allow an OPC UA Client to receive cyclic updates of OPC UA Variable values and Node attributes. A Subscription is the context needed to realize this cyclic exchange of information; MonitoredItems must be created inside a Subscription by the OPC UA Client and must be associated to OPC UA Nodes. The CreateSubscription and CreateMonitoredItem Services allow an OPC UA Client to create a subscription inside an existing Session and a MonitoredItem inside an existing Subscription, respectively [41].

MonitoredItems have several settings among which there is the SamplingInterval which defines the rate at which the OPC UA Server checks for changes in the associated Node, e.g. changes of the values for Variable Nodes and/or of the attributes for Object Nodes. If a change is detected, each MonitoredItem produces a particular message, called Notification, whose content depends on the changes detected; for example, in the case of changes of OPC UA Variable value, the parameter contains the new value

updated. Notifications are put in a queue defined inside each MonitoredItem. Size and queuing policy may be defined by the OPC UA Client for each MonitoredItem queue.

Each Subscription features a PublishingInterval, which defines the time interval at which the OPC UA Server clears all the MonitoredItem queues contained in the Subscription and conveys their contents (i.e. Notifications) into a NotificationMessage to be sent to the OPC UA Client. Transmission of NotificationMessages by OPC UA Server is triggered by a particular service called Publish exchanged between OPC UA Client and Server.

OPC UA specifications define a particular access control mechanism to the Nodes, based on the idea to separate authentication (determining who a client is) from authorization (determining what the Client is allowed to do). The access control features the concepts of role and permission. A role is a function allowed to a Client when it accesses a Server. For each role, a permission must be defined in OPC UA. RolePermissions is an optional attribute of BaseObjectType; it defines for a specific Node, the list of permission masks for each role. The permission mask specifies the allowed accesses to attributes of the Node (e.g., read, write, browse). RolePermissions is an array of RolePermissionType elements each one made up by the couple {role, permission}, specifying the permission mask available to a specific role [38].

3.2.2 oneM2M

In the oneM2M protocol, interaction with the resources are supported by the basic four CRUD (create, read, update, and delete) operations. According to the current version of the oneM2M specifications, CRUD operations may be realized for example by HTTP methods (e.g., GET, POST) [42], as it will be done in this proposal.

In order to understand how the access to resources by CRUD operations is realized in oneM2M, the following example will be given; it is based on a case study presented in [11].

Let us assume to have an IoT Server, an IoT embedded device with a temperature sensor and a smartphone application; IoT Server and smartphone application are realized by infrastructure nodes, whilst IoT embedded device is assumed to be realized by an application dedicated node. The AE residing in the IoT device (ADN-AE) must be registered to the IoT Server in order to be able to publish data; registration means that an <AE> type resource for the IoT device is created under the CSE resource tree of the IoT Server. In order to publish the values produced by the temperature sensor, a < container> resource is created under the <AE> type resource; furthermore, under this resource, one or more <contentInstance> resources are created, in which the sensing values of the temperature sensor are written. Figure 6.a shows on the left side, what just described.

The IN-AE residing in the smartphone application on the right side of Fig. 6.a is able to get the sensing values via oneM2M standard, according to two mechanisms: request/response and subscription/notification. According to the first one (marked by the circled number of '1' in Fig. 6.a), the IN-AE sends a HTTP GET request to the IoT Server with the URL (uniform resource locator) linked to the <contentInstance> resource it wants to get. If the request is asked with the appropriate privileges (i.e. according the access control policy contained in the ACP resource not shown in the figure), the IoT Server sends back the HTTP response containing the temperature sensor's value. The

second method (marked by the circled number of '2') can be accomplished by the common service function of the oneM2M platform, called 'subscription/notification'. The IN-AE creates a < subscription> resource under the container which it is interested in. Create <subscription> request service is used to create such resource [40]. On the basis of this subscription, the IoT Server will notify its subscriber (i.e. the IN-AE in the smartphone) of any events under the subscribed resource (e.g., new <contentInstance> resources are created under the <container> in order to publish novel sensing values of the temperature sensor). All these subscription/notification procedures will be performed by the HTTP POST Requests, as shown by Fig. 6.a.

Figure 6. b illustrates an actuation scenarios for IoT devices. The same IoT Server and smartphone application are present; an IoT embedded device with a with a lightbulb must be able to receive commands to switch on/off the lightbulb. Figure 6.b shows on the left side that the ADN-AE residing in the IoT device is registered to the IoT server, and a < container> resource is created under the <AE> for the light control. After that registration procedure, the IoT device will create a new <subscription> resource (marked by the circled number of '1') in order to get alerts as soon as new <contentInstance> resources will be created under the <container> (i.e., to be notified of control messages triggered by other applications). On the right side of Fig. 6.b, the smartphone IN-AE can send a control command to the lightbulb incorporated into the IoT device

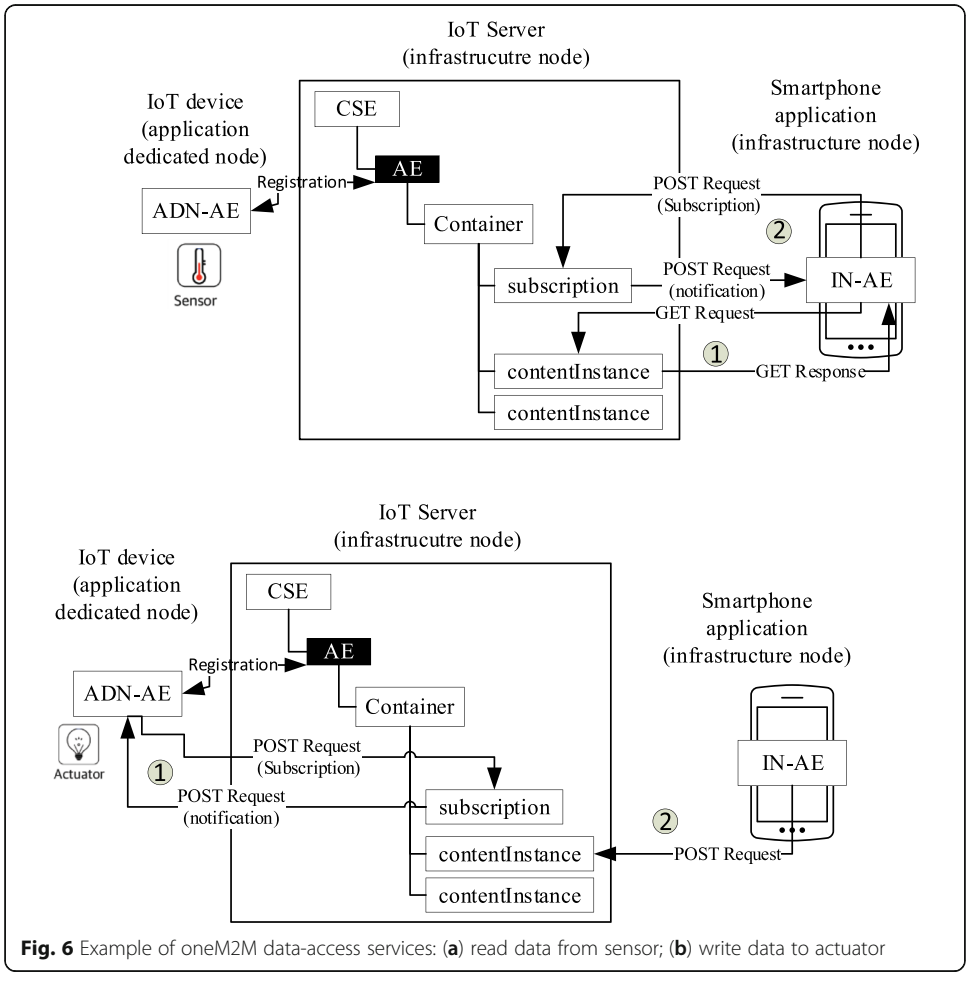

Fig. 6 Example of oneM2M data-access services: (**a**) read data from sensor; (**b**) write data to actuator

by creating a new <contentInstance> under the container for light control with the HTTP POST Request.

The oneM2M standard offers a resource discovery procedure allowing discovering of resources residing on a CSE resource tree. The use of the Filter Criteria parameter allows limiting the scope of the results. Resource discovery shall be accomplished by an Originator which shall also include the root of where the discovery begins (e.g. a resource of type <CSEBase>). An Originator could be an AE or another CSE. The unfiltered result of the resource discovery procedure includes all the child resources under the root of where the discovery begins, which the Originator has a Discover access right on [40].

In order to enhance interworking, oneM2M uses specialized interworking application entities called Interworking Proxy application Entity (IPE) [43]. IPEs are mainly characterized by two features: providing non-oneM2M reference points and remapping the related data model into the oneM2M-defined data model. An IPE is an AE that supports both oneM2M Mca reference point as well as the non-oneM2M interface, as shown by Fig. 7 [43].

4 Interoperability proposal

The interoperability proposal between OPC UA and oneM2M is based on the ad-hoc definition of an oneM2M IPE, which will be called OPCUA-IPE in the following. Figure 8 shows the OPCUA-IPE proposed. Two main entities are present: an OPC UA Server and the Interworking Manager.

Authors' aim is the integration of oneM2M–based IoT devices with OPC UA–compliant industrial applications; this integration requires that information produced by oneM2M–based IoT devices must be published by an OPC UA Server allowing the

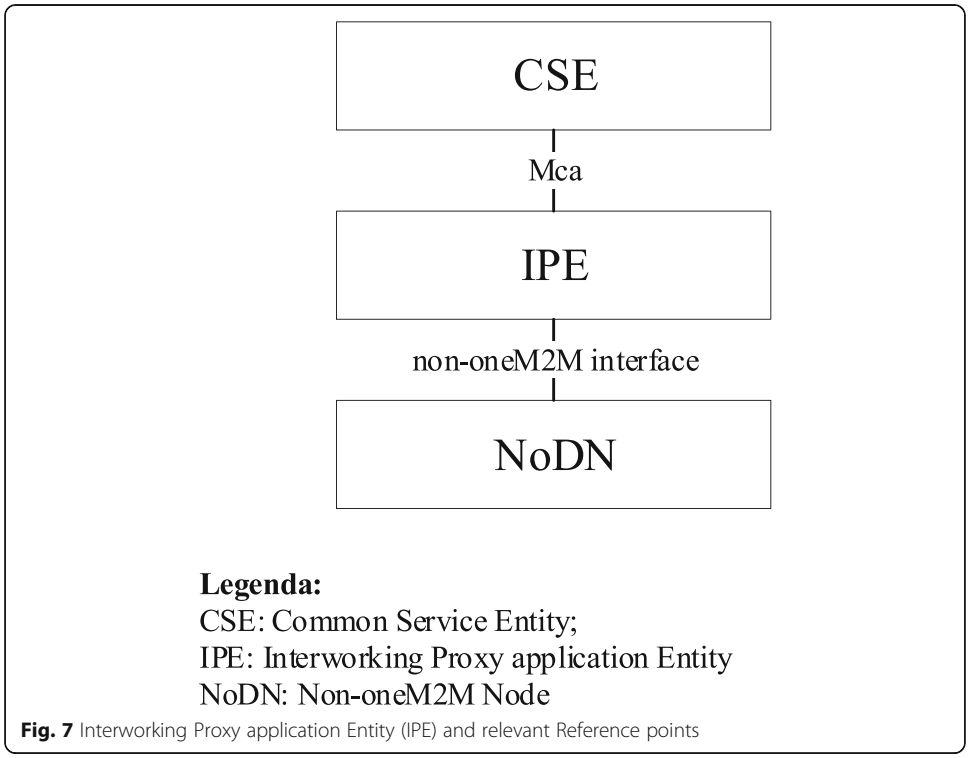

Fig. 7 Interworking Proxy application Entity (IPE) and relevant Reference points

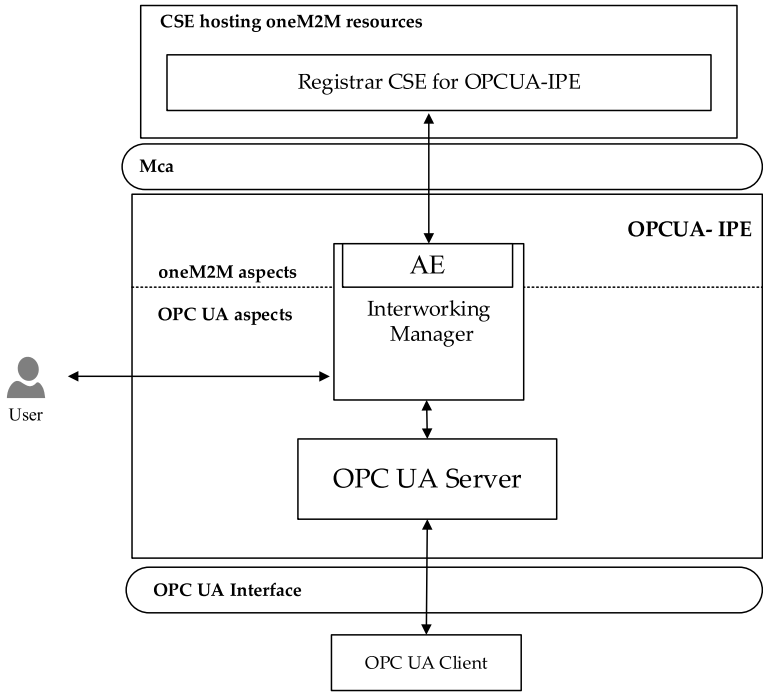

Fig. 8 OPCUA-IPE architecture

OPC UA-based client applications the access to this information. For this reason, the design of the OPCUA-IPE is based on the assumption to use an OPC UA Server to expose the resources belonging to the oneM2M system towards the OPC UA domain. The OPC UA Server contains the AddressSpace maintaining OPC UA Nodes mapping the oneM2M resources to be exposed towards the OPC UA domain. A mapping strategy between OPC UA and oneM2M Information Models has been defined by the authors to represent each oneM2M resource by a suitable set of standard or ad-hoc defined OPC UA Nodes inside the AddressSpace. The mapping procedure allows to set the attributes of each OPC UA Node according to the current value of the relevant oneM2M resource, represented by the Node. Each time a change occurs in an exposed oneM2M resource (e.g. updating of values of attributes), the change is reflected into the relevant set of OPC UA Nodes representing the oneM2M resource. In the opposite direction, each change inside the AddressSpace must be reflected in the correspondent oneM2M resource; for example if an OPC UA Client updates the attribute values of one or more OPC UA Nodes representing oneM2M resources, the relevant changes must be updated in these resources. The mapping strategy between OPC UA and oneM2M Information Models is fundamental in this proposal and it will clearly described in the following subsection.

The Interworking Manager is the core of the OPCUA-IPE. It communicates with the OPC UA Server and it is made up by an AE able to communicate with the CSE exposing the oneM2M resources mapped into the OPC UA domain. Section 5 will clearly point out the main activities performed by the Interworking Manager, among which there is the mapping of the OPC UA and oneM2M data access - based services described in Section 3.2.

It has been assumed that the Interworking Manager may receive inputs from a generic external user, as shown by Fig. 8; these inputs are limited to information about the choice of the oneM2M resources to be exposed and to be mapped into the OPC UA Server, as it will be pointed out by Section 5.

4.1 Mapping OPC UA and oneM2M information models

The main assumption taken for the definition of the OPCUA-IPE, is that the OPC UA AddressSpace of the Server must be populated in such a way that the oneM2M exposed resources are properly represented by the OPC UA Nodes. This requires a mapping process able to realize a one-to-one (or one-to-many if needed) correspondence between each oneM2M resource exposed and OPC UA elements. The authors realized this mapping process, which required the definition of novel elements in OPC UA (e.g. ObjectTypes, DataTypes), as the native ones were not able to represent the oneM2M resources.

Taking into account Fig. 4, it is important to recall that each oneM2M resource features both attributes and child-resources. As a general rule, it has been assumed to represent oneM2M attributes by OPC UA Variables of PropertyType or BaseDataVariableType. Properties are used to map the intrinsic characteristic attributes of resources that generally do not change value, or rarely do. DataVariables are considered to map attributes that change value frequently. The oneM2M child-resources have been represented as instances of ad-hoc OPC UA ObjectTypes which have been defined in the research carried out by the authors and will be described in the following.

An OPC UA ObjectType was defined and called ResourceObjectType in order to represent the basic structure of any oneM2M resource. Figure 9 shows the ResourceObjectType ObjectType, using OPC UA standard graphical representation. Properties of the ResourceObjectType have been defined to represent the attributes of the oneM2M resources. Figure 9 shows the properties representing the “universal attributes” described in Section 3.1.2; as shown by the figure, Mandatory ModellingRule Objects have been used for these properties as these attributes are mandatory.

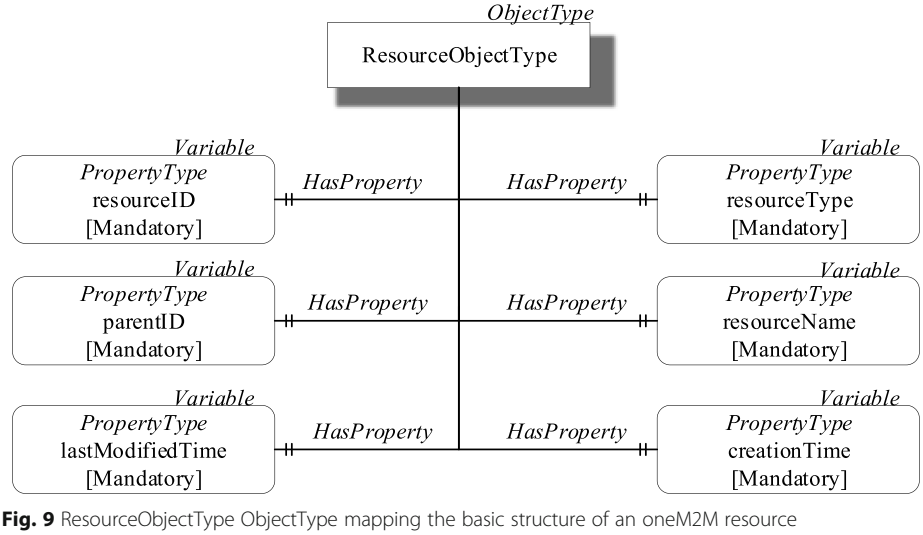

Fig. 9 ResourceObjectType ObjectType mapping the basic structure of an oneM2M resource

Other OPC UA ObjectTypes have been defined extending the ResourceObjectType ObjectType with the aim to realize the mapping of each oneM2M resource type. When extending the ResourceObjectType ObjectType, all the relevant properties are inherited and other Properties and/or Variables are added according to the particular attributes featured by the oneM2M resource to be represented; additional properties or variables are always added using the mechanism based on the OPC UA ModellingRule Object. Other ModellingRule Objects may be added in order to map the child-resources of the particular oneM2M resource type.

Figure 10 shows only the OPC UA ObjectTypes mapping oneM2M resources, which will be considered in the case study presented in this paper.

The cseBaseObjectType is an ObjectType representing the oneM2M <CSEBase> resource type. It extends ResourceObjectType adding other properties among which cseID and cseType (both mandatory). The cseID models the CSE-ID attribute containing the id of the oneM2M CSE. The cseType is another property mapping the oneM2M cseType attribute representing the type of CSE (e.g., IN-CSE or MN-CSE) [40]. The cseBaseObjectType may feature one or more Objects of aeObjectType

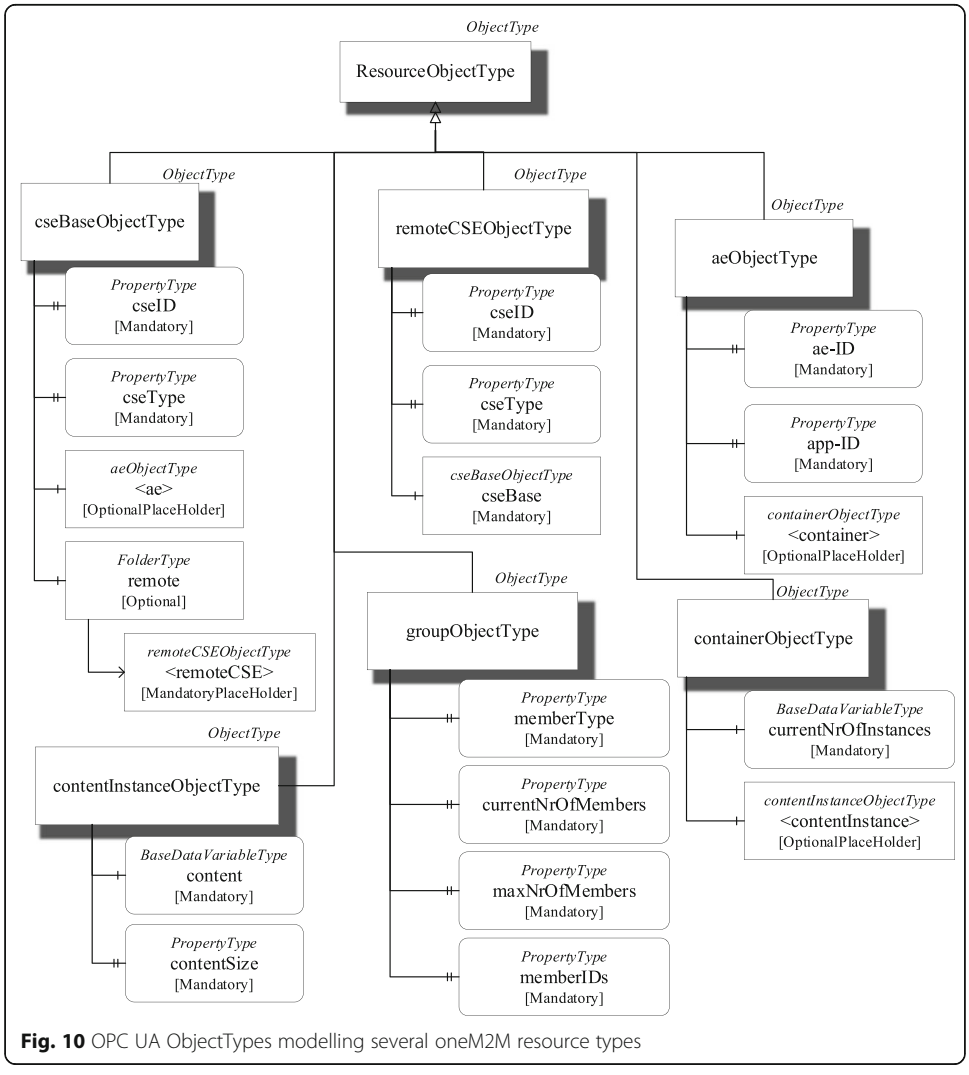

Fig. 10 OPC UA ObjectTypes modelling several oneM2M resource types

(described in the following) each of which represents an <AE> resource; for this reason, the <ae> Object featuring an OptionaPlaceHolder Modelling Rule is present as component. The cseBaseObjectType may also feature one or more Objects of remoteCSEObjectType (described in the following) each representing an oneM2M <remoteCSE> resource. A FolderType Object (named remote) has been defined to organize the remoteCSEObjetType Objects. If <remoteCSE> resources are registered in the Registrar CSE resource tree, the FolderType remote Object is present and, in this case, the presence of remoteCSEObjetType Objects modelling <remoteCSE> resources is mandatory; this explains the use of the MandatoryPlaceHolder ModellingRule associated to the <remoteCSE> Object, as shown by Fig. 10.

The remoteCSEObjectType is an ObjectType representing oneM2M <remoteCSE> resource type. Like cseBaseObjectType, it contains cseID and cseType properties. <remoteCSE> resource type also features the oneM2M CSEBase attribute, which represents the address of a < CSEBase> resource, relevant to the <remoteCSE> resource [40]. It has been represented in OPC UA as an instance of the cseBaseObjectType type, shown in Fig. 10 by the InstanceDeclaration cseBase Object. This choice allows to have all the child-resources of the <CSEBase>, to which the remoteCSE refers, allocated within the OPC UA Server.

The contentInstanceObjectType represents oneM2M <contentInstance> resource type. It holds a Variable as component, named content, allowing to represent what is contained in the contentInstance. Furthermore, it features a mandatory property named contentSize representing the size in bytes of the content Variable.

The containerObjectType represents <container> resource type. It includes two components: the mandatory variable currentNrOfInstance and OptionalPlaceholder contentInstance belonging to the contentInstanceObejctType described before.

The aeObjectType is an ObjectType representing oneM2M <AE> resource type. It extends ResourceObjectType adding ae-ID and app-ID as Mandatory Properties; furthermore, it optionally holds as component one or more container Objects belonging to the containerObjectType described before.

The groupObjectType represents one M2M <group> resource type. It features several properties. The memberType property has a Value belonging to the ad-hoc defined enumeration ResourceDataType allowing to specify what kind of resources are member of the group. The currentNrOfMembers property represents the current number of members, whose value cannot exceed the value of maxNrOfMembers (also shown by Fig. 10). Finally, there is the memberIDs property representing a collection of IDs of resources grouped.

The last question to point out is about how oneM2M authorization policy may be mapped into OPC UA. Section 3.1.2 pointed out that authorization policy is represented in oneM2M through the <accessControlPolicy> resource. The main assumption made in this proposal is that oneM2M authorization policy is realized through the OPC UA RolePermissions attribute which is present in each of the above defined ObjectTypes shown by Fig. 10. In particular, for each of the oneM2M resource linked to <accessControlPolicy> resource, the attribute RolePermissions of the OPC UA Node representing the oneM2M mapped resource, is set. The setting of this attribute must allow to specify the permissions to be applied for each role defined for the OPC UA clients and for each OPC UA Node as a pair {role, permission}. It is worth noting that

roles and permissions may be different in the oneM2M and OPC UA domains; the choice of roles and permissions in the OPC UA domain should be made in such a way to reflect the original ones, as faithfully as possible.

XML representation of the OPC UA ObjectTypes proposed for the oneM2M-OPC UA mapping has been realized by the authors on GitHub at the address [44]. The file named "onem2m-opcua.xml" is available and can be downloaded from this web site. This file has been prepared using the free tools UAModeler [45], which allow to build a customized OPC UA Information Model and to export it into XML format. The advantages to have an Information Model implemented into XML is that it can be easily imported into a generic OPC UA Server though the common OPC UA Software Development Kits (SDK) and libraries available in literature.

5 A detailed description of the OPCUA-IPE

The aim of this section is to give more details about the internal activities of the OPCUA-IPE. This description will allow to acquire a clear knowledge about the functionalities of each single component of the OPCUA-IPE.

5.1 Choice of the oneM2M resources to be exposed at start-up

Before the OPCUA-IPE may start its activity, suitable procedures must be performed at start-up with the aim to expose oneM2M resources towards OPC UA domain.

First of all, the AE inside the Interworking Manager must be registered in the CSE resource tree, otherwise any accesses to the relevant resources cannot be done. As defined in [39], the registration phase involves the creation of an <AE> resource in the CSE resource tree.

Once the registration procedure has been completed, the Interworking Manager has to create a < subscription> resource for each of the CSE resource to be exposed by the OPCUA-IPE. The subscription is needed in order the Interworking Manager could be notified about any changes which may occur to the relevant oneM2M resource (e.g. removal of the resource or update of some attributes of the resource); this kind of notification is very important as each change in oneM2M resources must be reflected into the AddressSpace of the OPC UA Server, as it will be explained in the following subsections.

In order the Interworking Manager could proceed to realize these subscriptions, it must be notified about which CSE resources must be exposed through the OPC UA Server. As pointed out in Section 4, it has been assumed that an external user should have the possibility to indicate the list of resources to be mapped inside the OPCUA-IPE. How an external user may give this information to the Interworking Manager may occur in several ways. The oneM2M standard does not provide any mechanisms for the definition of resources to be exposed by an IPE, but technical specifications [43, 46] suggest some methods to this aim; among them there are those called Pre-provisioning and Discovery. In the case of Pre-provisioning method, the list of the exposed resources is defined by the user through a preconfigured file. If Discovery method is used, the resources may be interactively chosen by the user, e.g. using a Graphical User Interface (GUI). The authors believe that adoption of these methods for the proposed OPCUA-

IPE is feasible. In the following, details of the procedures needed to realize them inside the OPCUA-IPE will be given.

If a Pre-provisioning method is used, the Interworking Manager receives by the user the list of resources to be exposed though a preconfigure file. In this case, a discovery process must be conducted by the Interworking Manager with the aim to confirm the presence of the requested resources in the Registrar CSE resource tree. If the resources exist, the Interworking Manager will proceed (through the local AE) to create a < subscription> resource for each of the resource contained in the list received, as said before. Figure 11 shows an example based on the Pre-provisioning method. In this case, the Interworking Manager is notified about the interest of the user to expose the two resources B and C (through a preconfigured file shown by the same figure). After the registration of the AE and after the discovery procedure, having had the confirmation about the existence of these resources in the Registrar CSE resource tree, the <subscription> resources for resources B and C are created. On the right side of Fig. 11, the new Registrar CSE resource tree is shown, after the creation of the <AE> and the <subscription> resources. This method has been chosen in the software implementation realized by the authors.

In the case of Discovery, the Interworking Manager will perform the discovery process with the aim to provide to the user the list of resources found; on the basis of this list, the user will have the opportunity to choose whether or not to expose each single resource. Once the Interworking Manager has received the list of resources to be exposed chosen by the user, it will proceed to create a < subscription> resource for each of the resource. Figure 12 shows an example of the Discovery method. As shown, the Interworking Manager gives to the user the list of resources found in the Registrar CSE through the discovery process. On the basis of this list, the user notifies the resources to be exposed at start-up (again B and C, like in the previous scenario). Finally, the Interworking Manager creates a < subscription> for each of these resources. On the right side of Fig. 12, the new Registrar CSE resource tree is shown, after the creation of the < AE> and the <subscription> resources.

Once the start-up phase here described has been concluded, the Interworking Manager will start the mapping of the resources specified by the user (i.e. {B,C} in the examples of Figs. 11 and 12) into OPC UA Server, as it will be explained in the following subsection.

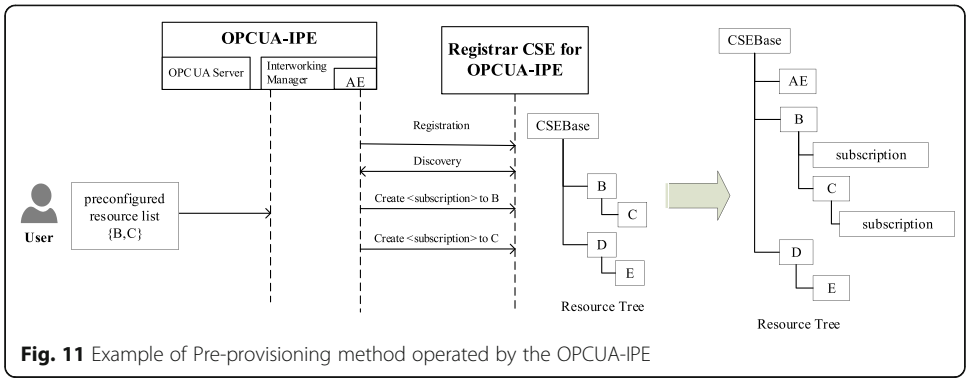

Fig. 11 Example of Pre-provisioning method operated by the OPCUA-IPE

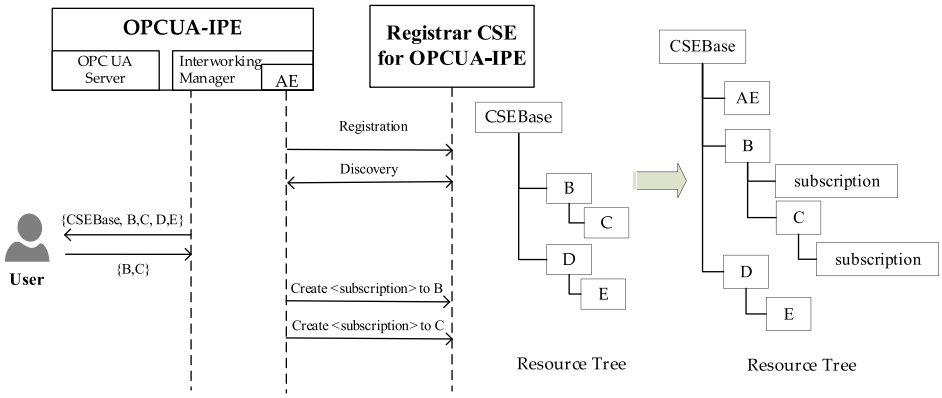

Fig. 12 Example of Discovery method operated by the OPCUA-IPE

5.2 OPC UA server creation and population

After the choice of the oneM2M resources to be exposed has been completed, the Interworking Manager will create an instance of the OPC UA Server inside the OPCUA-IPE.

Section 4 pointed out that the authors defined a mapping process able to realize a correspondence between each oneM2M resource and a suitable set of OPC UA Nodes. The "onem2m-opcua.xml" file containing the description of the mapping rules, is imported by the OPC UA Server in order to create inside the relevant AddressSpace the set of OPC UA types mapping the oneM2M resource types. This import operation is generally done by the same set of libraries available to build an OPC UA Server. For example, the FreeOpcUa libraries [47] used by the authors to implement the OPC UA Server contains the server.import_xml() method able to perform this task. From this moment on, the OPC UA Server will have knowledge of each OPC UA type defined for the mapping from oneM2M to OPC UA.

The next action performed by the Interworking Manager is the population of the OPC UA AddressSpace. For each oneM2M resource exposed (chosen at the start-up, as said before), the Interworking Manger is in charge to create an instance of the OPC UA type modelling the resource type. Once this instance has been created for a particular oneM2M resource, the attributes of the Nodes present in the instance must be filled with the same values coming from the oneM2M resource to be exposed.

Let us assume for example that the Interworking Manager has to map a < container> resource. Figure 13.a shows the exposed CSE resource to be mapped, including the values of the relevant attributes and the set of two <contentInstance> child-resources. Figure 13.b shows the instance of the OPC UA ObjectType containerObjectType (seen in Fig. 10), created inside the AddressSpace of the OPC UA Server. As it can be seen, the number of contentInstanceObjectType Objects created is two in order to map the two <contentInstance> child-resources. In the figure, the same values present in the attribute of the oneM2M resource are assigned to the relevant attributes of the OPC UA Nodes created inside the OPC UA Server.

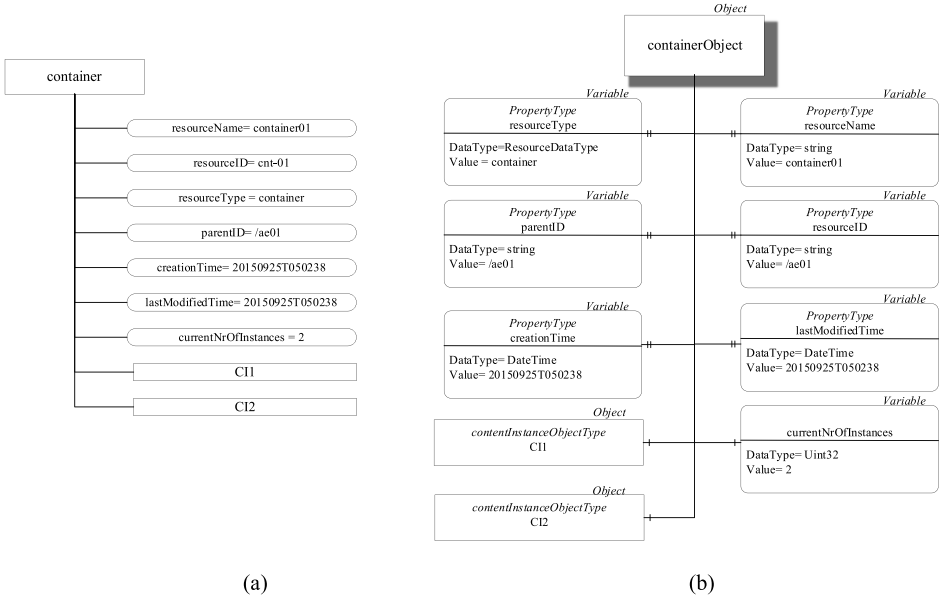

Fig. 13 Example of mapping a < container> resource, shown by (**a**), into the set of OPC UA Nodes shown by (**b**)

5.3 Interworking procedures

The aim of this subsection is that to detail the procedures adopted inside the OPCUA-IPE in order to realize the interworking, once the start-up phase has been concluded, the OPC UA Server has been instanced and the relevant AddressSpace has been populated.

The interworking procedures here defined have the aim to realize the mapping between OPC UA and oneM2M data access – based services described in Section 3.2. For each OPC UA service call performed by the client to access the information maintained by the AddressSpace of the OPC UA Server inside the OPCUA-IPE, the mapping with HTTP methods realizing the oneM2M CRUD operations with the CSE hosting one M2M resources must be realized, and vice versa.

Let us consider an OPC UA Client invoking the Read service on an OPC UA Node mapping an oneM2M resource. As said in Section 3.2.1, OPC UA Read Request features a maxAge parameter (expressed in milliseconds) specified by the OPC UA Client [41]. OPC UA specifications require that if the Server does not have a cached value which satisfies the requirements about the maximum age, it shall attempt to read a new value from the data source. Again, in the case the maxAge is set to 0, the Server shall attempt to read a new value from the data source. If maxAge is set to the max Int32 value or greater, the Server shall attempt to get a cached value. Finally, if the Server cannot meet the requested maxAge, it returns its “best effort” value rather than rejecting the request; this may occur when the time it takes the Server to process and return the new data value after it has been accessed from the data source is greater than the specified maximum age. It is clear that the data source is the Registrar CSE in this scenario. Figure 14 shows the complete interworking procedure used in the case the Server has to access the data source, i.e. the Registrar CSE; the figure points out the relevant information flow inside the OPCUA-IPE and between OPCUA-IPE and

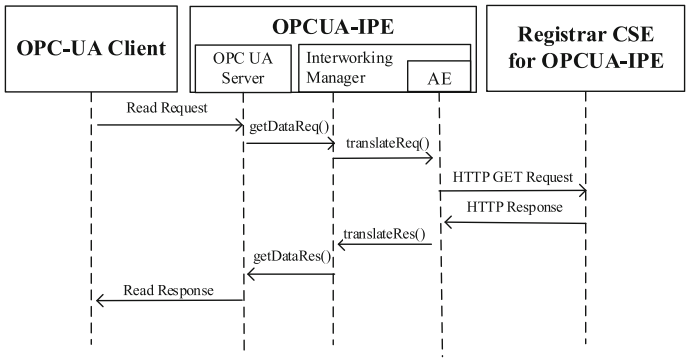

Fig. 14 Interworking procedure inside the OPCUA-IPE when a Read service is invoked by an OPC UA Client

Registrar CSE. For each Read Request received, the OPC UA Server will send an internal request to the Interworking Manager (i.e. getDataReq() in the figure); this request is aimed to access the oneM2M resource relevant to the OPC UA Node specified by the client in the Read Request. The Interworking Manager will prepare a request to its local AE, specifying the URI of the oneM2M resource to be accessed (i.e. translateReq() in the figure). The AE will issue a HTTP GET Request to access to the specified resource, as requested by [42]. Figure 14 shows the HTTP GET Request and the information flow in the opposite direction, when the HTTP Response is received. The data content of this service is forwarded to the OPC UA Server by the Interworking Manager (through the getDataRes(), as shown in the figure). The OPC UA Server will update the cached value of the relevant OPC UA Node involved in the previous Read Request and will send the requested attribute values to the OPC UA Client.

The information flow depicted in Fig. 14 for the Read Service may be used also for the Browse Service; the interworking procedure is very similar in this case and it will not be described for this reason.

Figure 15 shows the interworking procedure applied for each Write Request sent by an OPC UA Client to update a value of an OCP UA Node representing an oneM2M resource. The OPC UA Server will issue an internal request (i.e. writeDataReq()) to the Interworking Manager, which will request the AE the transmission of a HTTP POST Request, as requested by [42]. Also in this case, the Interworking Manager specifies the URI of the oneM2M resource to be accessed (i.e. translateReq() in the figure). On the receipt of the relevant HTTP Response, the Interworking Manager will confirm the writing operation to the OPC UA Server, by the writeDataRes(). The OPC UA Client will receive a Write Response sent by the OPC UA Server confirming its pending Write Request.

Interworking procedures must include also the scenario featured by the exchange of information based on OPC UA Subscription and MonitoredItems. Creation of Subscriptions and MonitoredItems by the OPC UA Client occurs according the OPC UA specifications [41] and it does not have an impact on the interworking process (i.e. it does not involve the Registrar CSE); for this reason, it will not consider in the following. Let us consider a Subscription already created inside the OPC UA Server and let us assume that it contains several MonitoredItems relevant to OPC UA Nodes mapping exposed oneM2M resources. According to the OPC UA specifications [41], each

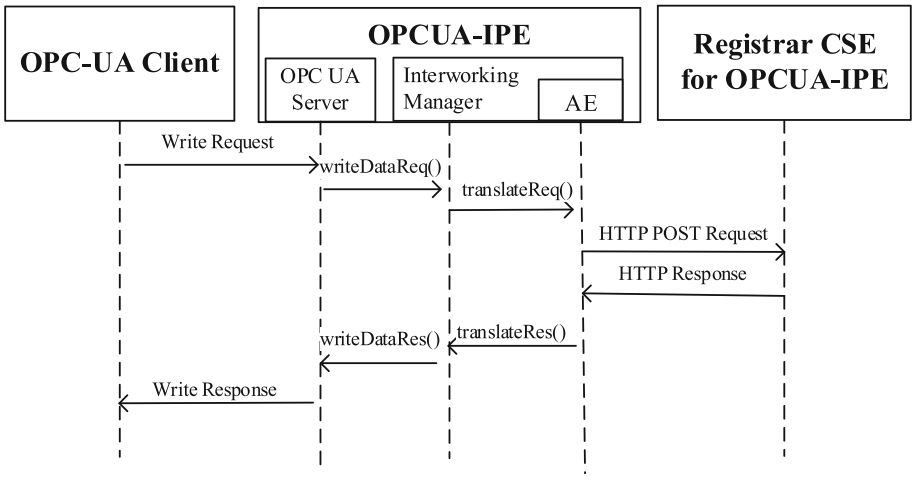

Fig. 15 Interworking procedure inside the OPCUA-IPE when a Write service is invoked by an OPC UA Client

MonitoredItem has to sample an attribute of an OPC UA Node each time the SamplingInterval elapses. Figure 16 shows the procedure to be adopted. Each time the SamplingInterval elapses, a getDataReq() is sent by the OPC UA Server to the Interworking Manager to request the access to the original oneM2M resource related to the OPC UA Node linked to the MonitoredItem. As done for the Read service, a HTTP GET Request is used to access to the oneM2M resource. Once a value is received by a HTTP Response, it is sent to the MonitoredItem to be enqueued in the relevant queue, according to the OPC UA specifications [41]. As said in Section 3.2.1, OPC UA Client will receive the values enqueued for each MonitoredItem and each Subscription by OPC UA Notification messages, as shown by Fig. 16.

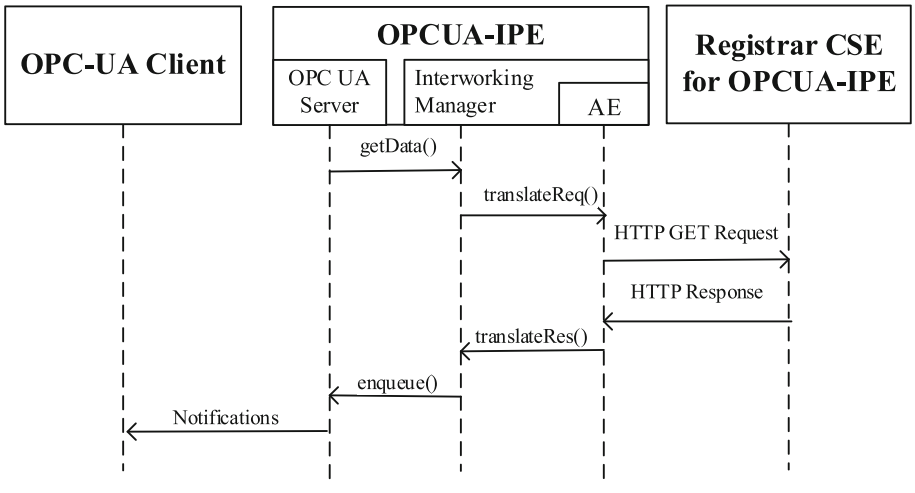

Fig. 16 Interworking procedure inside the OPCUA-IPE considering the exchange of information based on OPC UA Subscription and Monitored Items

5.4 Updating exposed oneM2M resources

At run-time changes in the Registrar CSE resource tree may happen; for example, exposed resource are removed, or attributes of exposed resources are updated. It may also happen that other oneM2M resources must be exposed in the AddressSpace of OPC UA Server, e.g. if new resources are available in the Registrar CSE resource tree. It is required that the Interworking Manager must be in charge to take into proper account these changes. In the following, an analysis of the main changes and the relevant actions carried out by the Interworking Manager will be clearly pointed out.

Let us consider the exposed oneM2M resources. It is important to recall that at start-up phase, a < subscription> resource has been created for each of the exposed resource (as shown by Fig. 11 and Fig. 12, for example). This means that the AE inside the Interworking Manager receives a notification message each time a change in one of the subscribed-to oneM2M resources occurs. Notifications sent to the AE are generated depending on the eventNotificationCriteria set chosen for each <subscription> resource [39]. In this paper it has been assumed that the notification criteria include all the possible updates to a specific resource, foreseen by the oneM2M standard; in particular they are relevant to: update to attributes of the subscribed-to resource, deletion of the subscribed-to resource, creation of a child-resource of the subscribed-to resource, deletion of a child-resource of the subscribed-to resource.

Interworking Manager inside the OPCUA-IPE triggers notifications received by the Registrar CSE and makes the relevant update in the AddressSpace of the OPC UA Server. Figure 17 points out the information flow occurring in this case; the notify() message is sent by the local AE to the Interworking Manager for each notification message received from Registrar CSE. The update() message is sent to the OPC UA Server as a consequence in order to realize the relevant update.

Update of the AddressSpace depends on the cause of the notification, off curse. If an update to the attributes of exposed oneM2M resources occurred, the update to the relevant attributes of the mapping OPC UA Nodes are realized. If a deletion of the subscribed-to resource occurred, the relevant OPC UA Nodes are removed from the AddressSpace. In the case of creation of a child-resource of the subscribed-to resource, the OPC UA Nodes representing the child-resource are added into the AddressSpace.

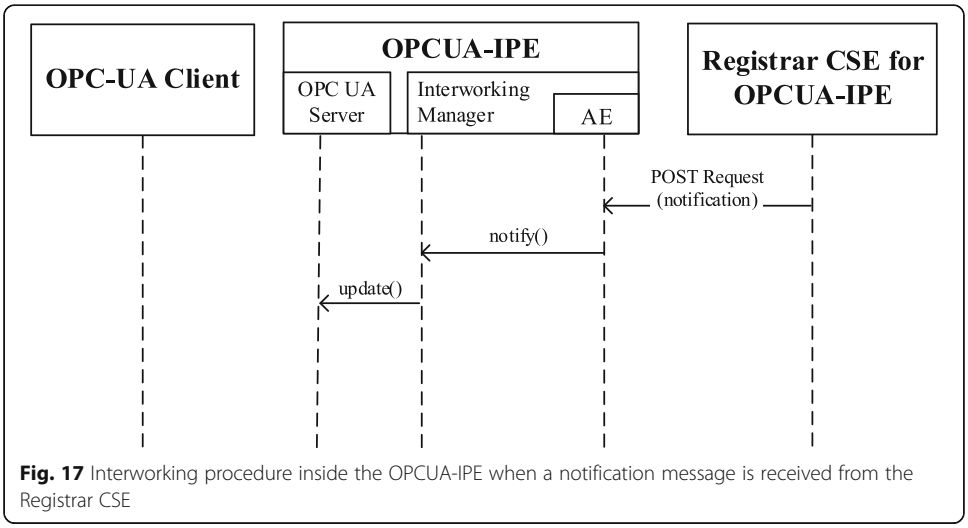

Fig. 17 Interworking procedure inside the OPCUA-IPE when a notification message is received from the Registrar CSE

Finally, in the case of deletion of a child-resource of the subscribed-to resource, the relevant OPC UA Nodes are removed.

Let us consider another scenario featured by the need to add one or more oneM2M resources at run-time. This may occur when new oneM2M resources are added inside the Registrar CSE resource tree and the user wants to expose them. The scenario may also occur when the user requires to expose oneM2M resources already existing (e.g. at start-up phase) and not previously exposed. In order to handle this scenario, it has been assumed to adopt another method for the definition of resources to be exposed by an IPE suggested by the technical specifications [43, 46]; it is called On-demand Discovery. On-demand Discovery allows the dynamically update of the list of resources to be exposed, according to the user's choice (like the Discovery method). It has been assumed that at run-time, the Interworking Manager repeats the discovery procedure at certain intervals. In this way, it can achieve an update list of the available oneM2M resources, including new oneM2M resources added to the Registrar CSE resource tree; it will communicate this list to the user, in order this last could notify the list of resources to be exposed by the OPCUA-IPE. Adding new exposed oneM2M resources implies the same procedure described in subsection 5.1 (i.e. creation of the <subscription> resource for the added oneM2M resource) and those described in subsection 5.2 (i.e. creation of the OPC UA Nodes inside the OPC UA Server representing the oneM2M resource to be exposed at run-time).

6 Case study

The aim of this section is that to give an example of application of the interworking proposal just presented. The same case study presented in [48] will be taken into consideration. Although the example does not refer to an industrial scenario, it is easy to be understood and it allows a better understanding of the proposal here presented.

The scenario deals with a home lighting system that can be remotely controlled by a user's smartphone, using the oneM2M architecture. A graphical overview of the case study is shown by Fig. 18. The lighting system is deployed in a home and is attached to a Home Gateway. The gateway communicates with a cloud service platform in the infrastructure domain, allowing the lights to be remotely controlled by the smartphone shown by the figure. The cloud service platform supports a set of services to allow the smartphone the control of the lights in the home; to reach this aim, the smartphone hosts an application used for the actual control.

In this case study, the smartphone and each of the devices inside the lighting system hosts an AE, as shown in the figure. ADN-AE1 and ADN-AE2 are applications embedded in Light1 and Light2 devices at the field domain, respectively; they have the capabilities to actuate the control commands sent to the lighting system. IN-AE inside the smartphone is an embedded application with capabilities to interact directly with the oneM2M cloud service platform IN-CSE. Two CSEs are present in the system. An IN-CSE is hosted in the cloud by the oneM2M Service Provider and a MN-CSE is hosted on the Home Gateway. IN-CSE located in the Cloud Service Platform interacts with MN-CSE allowing the remote control of Light1 and Light2 devices by the smartphone, through the IN-CSE. A Mca reference point is used between each Light AE and MN-CSE Home Gateway and between Smartphone AE and the Cloud Service Platform IN-CSE; Mcc is the reference point used between the Home Gateway MN-CSE and

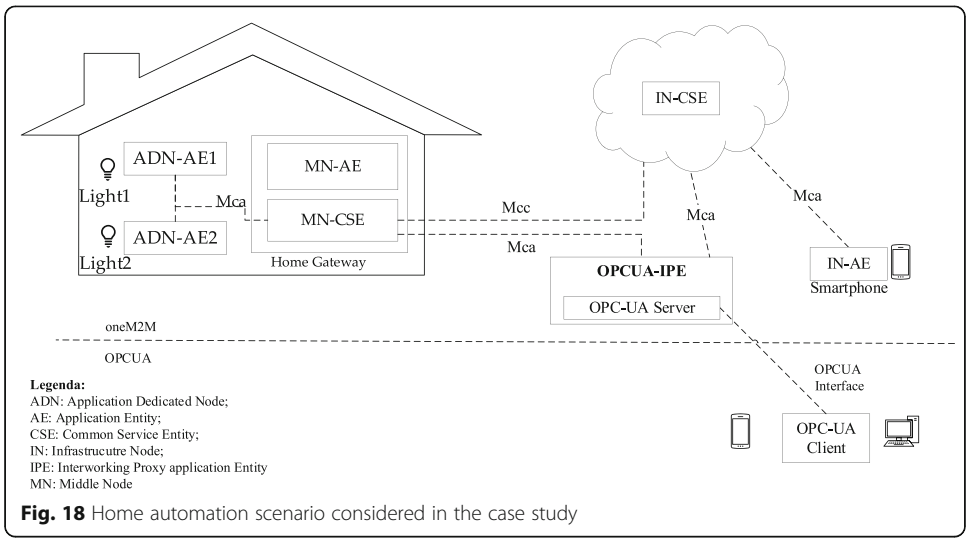

Fig. 18 Home automation scenario considered in the case study

oneM2M service platform IN-CSE. Finally, MN-AE is a gateway application embedded into the home gateway that interacts with the MN-CSE through Mca reference point.

Figure 19 shows the resource tree assumed for the Home Gateway MN-CSE, chosen on the basis of the case study presented in [48].

It starts with a < CSEBase> resource named home_gateway and is made up by the "universal attributes" and by the child-resources described in the following. There is an < accessControlPolicy> resource named gateway_acp; the two <AE> resources named light_ae1 and light_ae2 refer to the two lighting systems shown by Fig. 18. It has been assumed that the first lighting system (i.e. Light1) features only one data instance, whilst Light2

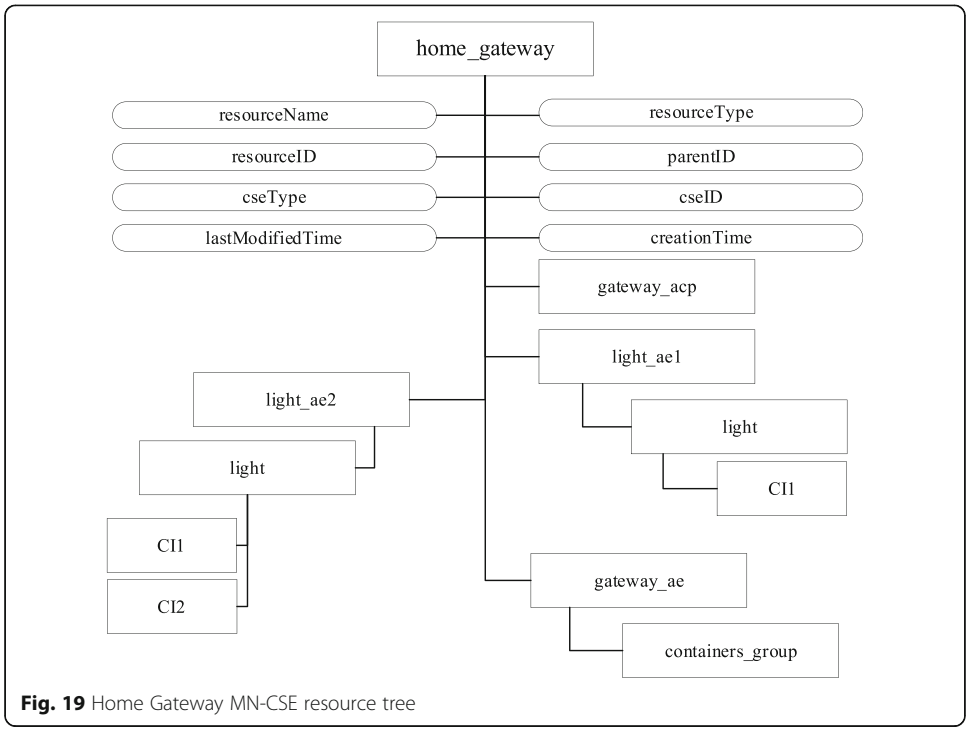

Fig. 19 Home Gateway MN-CSE resource tree

features two data instances. For this reason, light_ae1 contains a < container> sub-resource named light, featuring only a < contentInstance> resource, named CI1. The other <AE> resource named light_ae2, contains another <container> sub-resource named light, featuring two <contentInstance> resources, named CI1 and CI2. Finally, there is an <AE> resource named gateway_ae which contains a < group> resource named containers_group whose members are the light containers of each ADN-AEs.

As described in [48], the oneM2M resource tree of IN-CSE starts with a < CSEBase> resource named Server, and two child-resources, a < remoteCSE> named home_gateway and an <AE> named smartphone_ae. The home_gateway resource contains the address of the <CSEBase> resource represented by this <remoteCSE>, i.e. the address of the home_gateway <CSEBase> resource contained in the Home Gateway MN-CSE, and previously described.

Figure 18 points out the presence of the OPCUA-IPE proposed in this paper, to realize interworking from oneM2M to OPC UA. Taking into account the case study analyzed here, OPCUA-IPE may be connected to IN-CSE of the infrastructure domain and/or to the MN-CSE of the Home Gateway in the field domain. Mca reference point is used in both cases, as shown by Fig. 18. In this case study, it has been assumed to connect the OPCUA-IPE to the MN-CSE. Due to this assumption, the OPC UA Server inside the OPCUA-IPE will include only the OPC UA Nodes representing the oneM2M resources shown by Fig. 19. Figure 20 shows the mapping of the Home Gateway MN-CSE resource tree to the AddressSpace of the OPC UA Server inside the OPCUA-IPE. The figure shows the OPC UA Nodes used in the mapping, giving few details in terms of their attributes only due to the lack of space. The mapping has been realized according the process described in Section 4.1.

HomeGateway Object is an instance of cseBaseObjectType, mapping the home_gateway <CSEBase> resource. This Object features two components; they are the two instances of aeObjectType named light_ae1 and light_ae2, mapping the relevant <AE> resources. The Objects named light map the relevant <container> resources; the Objects CI1 and CI2 map the <contentInstance> resources with the same name. As said in Section 4.1, the <gateway_acp > resource is not mapped into OPC UA Server, as this mapping is realized by the definition of suitable roles and permissions inside the server in order to model the attribute privileges of the <gateway_acp > resource.

7 Final remarks

The paper has presented an interworking proposal between OPC UA and oneM2M, which enables the access of information maintained by oneM2M-based systems/platforms to applications based on OPC UA. This has the advantage to enhance interoperability inside Industry 4.0 where a very strong requirement is the integration of industrial applications with the IoT domains. The authors believe that interworking between OPC UA and oneM2M protocols is important as both of them are considered strategic communication frameworks in Industry 4.0 reference architectures. The paper is original as no other contributions are present in the current literature with the same subject.

The interworking solution presented in this paper is based on an oneM2M IPE called OPCUA-IPE; the internal architecture has been ad-hoc defined according the aim of the paper. The OPCUA-IPE has been implemented by authors and the source code is

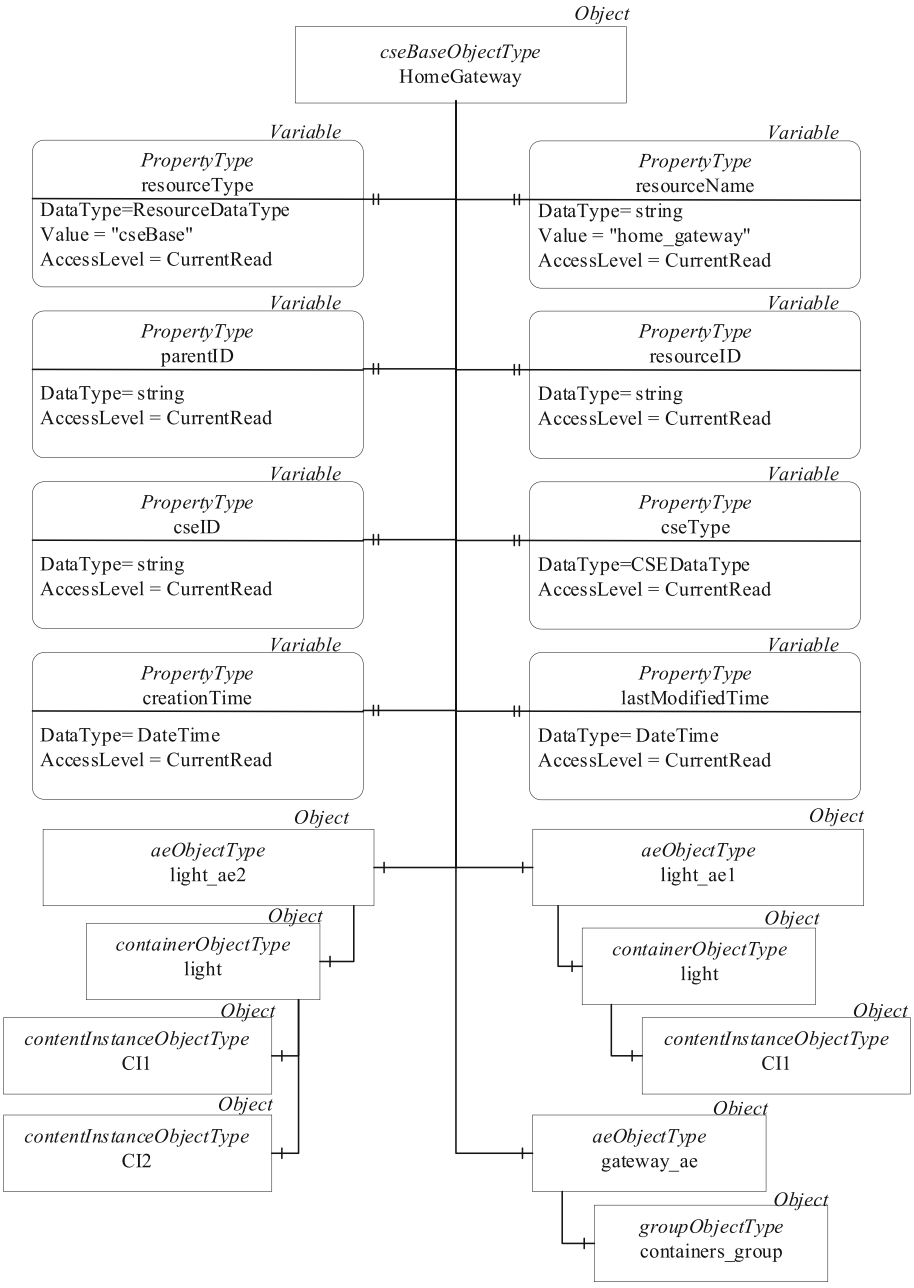

Fig. 20 OPC UA Nodes mapping the Home Gateway MN-CSE resource tree

freely available on GitHub [49]. Implementation was based on Phyton language, chosen mainly to allow an easier integration with libraries and SDK used to realize some elements of the OPCUA-IPE architecture. Implementation of the AE inside the Interworking Manager has been based on OpenMTC SDK [50]. OpenMTC is a python-based reference implementation of the oneM2M standard. A new class has been created extending from the base class XAE [50], which is in charge to provide resource discovery, subscription and resource management. Extension of this class realized by the authors mainly allow the AE to perform several activities of interworking procedure described in Section 5; the main activities implemented are: registration of AE to the CSE,

discovery of the entire resource tree of the Registrar CSE, creation of <subscription> resources inside the Registrar CSE resource tree, management of each notification received from the registered resources of the Registrar CSE (see notify() in Fig. 17), management of traslateReq() and traslateRes() internal services exchanged between Interworking Manager and AE and their mapping to the HTTP POST and GET exchanged with the CSE (see Figs. 14, 15 and 16). Interworking Manager is the core of the architecture. It has been developed in Phyton as a new class entirely defined by the authors. This class is in charge to create an instance of OPC UA Server, to populate the AddressSpace with the Nodes mapping oneM2M resources, to realize the translateReq() and translateRes() internal services, and to realize the updates the AddressSpace for each notification received from the AE. Implementation of OPC UA Server is based on FreeOpcUa [47], a python based open-source OPC UA communication stack. Furthermore, UaExpert OPC UA Client [51] was used during the test carried out by the authors. Some classes of the FreeOpcUa libraries have been extended by the authors. In particular the internal services getDataReq(), getDataRes(), writeDataReq(), writeDataRes(), getData(), enqueue(), update() (see Figs. 14, 15, 16 and 17), were defined through the new classes extending the basic ones. These services reflect the interworking procedures described in Section 5.3.

Abbreviations

ADN: Application Dedicated Node; AE: Application Entity; CRUD: Create, Retrieve, Update, and Delete; CSE: Common Service Entity; DPWS: Device Profile for Web Services; GUI: Graphical User Interface; IIC: Industrial Internet Consortium; IIRA: Industrial Internet Reference Architecture; IN: Infrastructure Node; IoT: Internet of Things; IPE: Interworking Proxy application Entity; M2M: machine-to-machine; MN: Middle Node; NoDN: Non-oneM2M Node; NSE: Network Services Entity; OPC UA: Open Platform Communications Unified Architecture; RAMI 4.0: Reference Architecture Model for Industry 4.0; ROA: Resource-Oriented Architecture; SDK: Software Development Kits (SDK); URI: Uniform Resource Identifier

Acknowledgments

Not applicable.

Authors' contributions

The authors equally contributed to the elaboration of this paper. All the authors read and approved the final manuscript.

Authors' information

Salvatore Cavalieri was born in Catania (Italy) in 1965. He received his "laurea" degree in Electronic Engineering from the University of Catania in 1989. In 1993 and 1995, he received a Ph.D. in Electronic and Computer Engineering, and a post-PhD in Electrical Engineering from the same University. Currently he is Full Professor of Computer Engineering at the University of Catania, Department of Electrical Electronic and Computer Engineering. His main research areas are in distributed systems, real-time scheduling, industrial informatics, IoT, IIoT, and process control-oriented communication protocols. Salvatore Mulè was born in Leonforte (Italy) in 1991. He studied at the University of Catania, where he obtained his Master's Degree in Computer Science in 2019. He is currently a Ph.D. student in Systems, Energy, Computer and Telecommunications Engineering at University of Catania. His current research includes Industry 4.0, IoT, IIoT. He is co-author of several papers about definition of Interoperability solutions for Industry 4.0 and IIoT, based on OPC UA and oneM2M standards.

Funding

This work was supported in part by the "Piano di incentivi per la ricerca di Ateneo 2020/2022 (PIA.CE.RI.)", University of Catania.

Availability of data and materials

The results of the paper involved the definition of XML representation of the OPC UA Nodes proposed in the paper for the mapping between oneM2M and OPC UA. This definition is maintained at *https://github.com/OPCUAUniCT/oneM2M-to-OPCUA-Information-Models-mapping* .
Furthermore, the IPE architecture presented in the paper and called OPCUA-IPE has been fully implemented and the source code is maintained at *https://github.com/OPCUAUniCT/oneM2M-OPCUA-IPE* .
Both repositories are downloadable under the Apache 2.0 license.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 31 May 2020 Accepted: 14 November 2021
Published online: 06 December 2021

References

1. Alam M, Nielsen RH, Prasad NR. The Evolution of M2M into IoT. In: Proceedings of First International Black Sea Conference on Communications and Networking (BlackSeaCom); 2013. p. 112–5.
2. Guarda T, Leon M, Augusto MF, Haz L, de la Cruz M, Orozco W, et al. Internet of Things Challenges. In: Proceedings of 12th Iberian Conference on Information Systems and Technologies (CISTI). Lisbon: IEEE Computer Society; 2017.
3. Porter ME, Heppelmann JE. How smart, connected products are transforming competition. Harv Bus Rev. 2014;92:64–88.
4. Wang S, Wan J, Li D, Zhang C. Implementing Smart Factory of Industrie 4.0: An Outlook. Int J Distrib Sens Netw. 2016; 12:1–10.
5. Liao Y, Deschamps F, Loures EFR, Ramos LFP. Past, present and future of industry 4.0-a systematic literature review and research agenda proposal. Int J Prod Res. 2017;55(12):3609–29. https://doi.org/10.1080/00207543.2017.1308576.
6. Xu LD, Xu EL, Li L. Industry 4.0: State of the art and future trends. Int J Prod Res. 2018;56:2941–62.
7. Swetina J, Lu G, Jacobs P, Ennesser F, Song J. Toward a standardized common M2M service layer platform: introduction to oneM2M. IEEE Wirel Commun. 2014;21(3):20–6. https://doi.org/10.1109/MWC.2014.6845045.
8. Alaya MB, Banouar Y, Monteil T, Chassot C, Drira K. OM2M: extensible ETSI-compliant M2M service platform with self-configuration capability. Procedia Comput Sci. 2014;32:1079–86. https://doi.org/10.1016/j.procs.2014.05.536.
9. Yun J, Ahn IY, Sung NM, Kim J. A device software platform for consumer electronics based on the internet of things. IEEE Trans Consum Electron. 2015;51(4):564–71. https://doi.org/10.1109/TCE.2015.7389813.
10. Kim J, Choi SC, Ahn IY, Sung NM, Yun J. From WSN towards WoT: open API scheme based on oneM2M platforms. Sensors. 2016;16(10):1645. https://doi.org/10.3390/s16101645.
11. Yun J, Ahn IY, Song J, Kim J. Implementation of sensing and actuation capabilities for IoT devices using oneM2M platforms. Sensors. 2019;19(20):4567. https://doi.org/10.3390/s19204567.
12. Yun J, Ahn IY, Choi SC, Kim J. TTEO (things talk to each other): programming smart spaces based on IoT systems. Sensors. 2016;16(4):467. https://doi.org/10.3390/s16040467.
13. Sicari S, Rizzardi A, Coen-Porisini A, Grieco LA, Monteil T. Secure OM2M service platform. In: Proceedings of the IEEE International Conference on Autonomic Computing (ICAC). Grenoble: IEEE Computer Society; 2015. p. 313–8.
14. Ryu M, Yun J, Miao T, Ahn IY, Choi SC, Kim J. Design and Implementation of a Connected Farm for Smart Farming System. In: Proceedings of the IEEE Sensors. Busan: IEEE Press; 2015. p. 1724–8.
15. Ryu M, Kim J, Yun J. Integrated semantics service platform for the internet of things: A case study of a smart office. Sensors. 2015;15(1):2137–60. https://doi.org/10.3390/s150102137.
16. Fattah SMM, Sung NM, Ahn IY, Ryu M, Yun J. Building IoT Services for Aging in place using standard-based IoT platforms and heterogeneous IoT products. Sensors. 2017;17(10):2311. https://doi.org/10.3390/s17102311.
17. Kovacs E, Bauer M, Kim J, Yun J, Gall FL, Zhao M. Standards-based worldwide semantic interoperability for IoT. IEEE Commun Mag. 2016;54(12):40–6. https://doi.org/10.1109/MCOM.2016.1600460CM.
18. An J, Gall FL, Kim J, Yun J, Hwang J, Bauer M, et al. Toward global IoT-enabled smart cities interworking using adaptive semantic adapter. IEEE Internet Things J. 2019;6(3):5753–65. https://doi.org/10.1109/JIOT.2019.2905275.
19. Zhao R, Wang L, Zhang X, Zhang Y, Wang L, Peng H. A OneM2M-compliant stacked middleware promoting IoT Research and Development. IEEE Access. 2018;6:63546–59. https://doi.org/10.1109/ACCESS.2018.2876197.
20. Wiramaswara Widya P, Yustiawan Y, Kwon J. A oneM2M-based query engine for internet of things (IoT) data streams. Sensors. 2018;18(10):3253. https://doi.org/10.3390/s18103253.
21. Weyer S, Schmitt M, Ohmer M, Gorecky D. Towards industry 4.0-standardization as the crucial challenge for highly modular, multi-vendor production systems. IFAC-PapersOnLine. 2015;48(3):579–84. https://doi.org/10.1016/j.ifacol.2015.06.143.
22. German Institute for Standardisation. Reference Architecture Model Industrie 4.0 (RAMI4.0), DIN SPEC 91345, 2016. Deutsches Institut für Normung [DIN].
23. Industrial Internet Consortium. The Industrial Internet of Things Volume G1: Reference Architecture (Version 1.80). https://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf. Accessed 12 Sep 2020. Industrial Internet Consortium (IIC) Technology Working Group and its Architecture Task Group.
24. Industrial Internet Consortium. The Industrial Internet of Things Volume G5: Connectivity Framework (Version 1.01). https://www.iiconsortium.org/pdf/IIC_PUB_G5_V1.01_PB_20180228.pdf. Accessed 12 Sep 2020. Industrial Internet Consortium (IIC) Technology Working Group and its Architecture Task Group.
25. Mahnke W, Leitner SH, Damm M. OPC unified architecture. Berlin Heidelberg: Springer-Verlag; 2009. https://doi.org/10.1007/978-3-540-68899-0.
26. Candido G, Jammes F, de Oliveira JB, Colombo AW. Soap at device level in the industrial domain: Assessment of OPC UA and DPWS specifications. In: Proceedings of 8th IEEE International Conference on Industrial Informatics; 2010. p. 598–603.
27. Sauter T, Lobashov M. How to access factory floor information using internet technologies and gateways. IEEE Trans Ind Inform. 2011;7(4):699–712. https://doi.org/10.1109/TII.2011.2166788.
28. Izaguirre M, Martinez Lastra JL, Lobov A. OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing. In: Proceedings of 9th IEEE International Conference on Industrial Informatics. Lisbon: IEEE Industrial Electronics Society; 2011. p. 205–11.
29. Grüner S, Pfrommer J, Palm F. A restful extension of OPC UA. In: Proceedings of IEEE World Conference on Factory Communication Systems (WFCS 2015); 2015. p. 1–4.
30. Grüner S, Pfrommer J, Palm F. Restful industrial communication with OPC UA. IEEE Trans Ind Inform. 2016;12(5):1832–41. https://doi.org/10.1109/TII.2016.2530404.
31. Wang P, Pu C, Wang H. OPC UA message transmission method over CoAP 01. In: . https://tools.ietf.org/html/draft-wang-core-opcua-transmission-01. Accessed 12 Sep 2020. Internet Engineering Task Force (IETF).

32. Derhamy H, Rönnholm J, Delsing J. Protocol interoperability of OPC UA in service oriented architectures. In: Proceedings of 15th IEEE International Conference on Industrial Informatics (INDIN). Emden: IEEE Industrial Electronics Society; 2017. p. 44–50.
33. oneM2M. TR-0018-V-4.0.0: Industrial Domain Enablement. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=29334 Accessed 12 Sep 2020. Technical Report published by onM2M organisation.
34. Lai P, Lin FJ. Industrial Internet of Things Interoperability Between OPC UA and OneM2M. IoT as a Service. In: Li B, Zheng J, Fang Y, Yang M, Yan Z, editors. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 316. Cham: Springer; 2020. p. 439–55.
35. Cavalieri S, Mulè S, Salafia MG. Enabling OPC UA and oneM2M Interworking. In: Proceedings of IEEE International Conference on Industrial Technologies (ICIT 2020), vol. 2020. Buenos Aires: IEEE Industrial Electronics Society.
36. Cavalieri S, Mulè S. Towards Interoperability of oneM2M and OPC UA. In: Proceedings of International Conference on Enterprise Information System (ICEIS 2020). Prague (Czech Republic): SCITEPRESS; 2020.
37. Pras A, Schoenwaelder J. On the difference between InformationModels and data models. In: Internet Engineering Task Force, RFC, vol. 3444; 2003. http://www.rfc-editor.org/rfc/rfc3444.txt. Accessed 12 Sep 2020.
38. OPCFoundation. Part 3: Address Space Model. https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model. Accessed 12 Sep 2020.
39. oneM2M. TS-0001-V4.5.0: Functional Architecture. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=31839 Accessed 12 Sep 2020.
40. oneM2M. TS-0004-V4.0.0: Service Layer Core Protocol. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=31772 Accessed 12 Sep 2020.
41. OPCFoundation. Part 4: Services. https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-services/ Accessed 12 Sep 2020.
42. oneM2M. TS-0009-V3.5.0: HTTP Protocol Binding. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=31202 Accessed 12 Sep 2020.
43. oneM2M. TS-0033-V3.0.0: Interworking Framework. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=29581 Accessed 12 Sep 2020.
44. oneM2M-to-OPCUA-Information-Models-mapping. https://github.com/OPCUAUniCT/oneM2M-to-OPCUA-Information-Models-mapping. Accessed 12 Sep 2020. GitHub.
45. UaModeler. https://www.unified-automation.com/products/development-tools/uamodeler.html. Accessed 12 Sept 2020.
46. oneM2M. TS-0024 V3.2.2: OCF Interworking. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=29565 Accessed 12 Sep 2020.
47. FreeOpcUa. https://github.com/FreeOpcUa/python-opcua. Accessed 12 Sep 2020. GitHub.
48. oneM2M. TR-0025 V2.0.3: Application Developer Guide. http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=31000 Accessed 12 Sep 2020.
49. oneM2M-OPCUA-IPE. https://github.com/OPCUAUniCT/oneM2M-OPCUA-IPE. Accessed 12 Sep 2020. GitHub.
50. OpenMTC. https://www.openmtc.org/index.html. Accessed 12 Sep 2020.
51. UaExpert-A Full-Featured OPC UA Client. https://www.unified-automation.com/products/development-tools/uaexpert.html. Accessed 12 Sep 2020.

Publisher's Note