ONEM2M
TECHNICAL SPECIFICATION

Document Number	TS-0033-V3.0.0
Document Name:	Interworking Framework
Date:	2019-04-03
Abstract:	This document is the specification describing interworking methodologies that are defined by oneM2M for the purpose of representing interactions with devices or functions in Proximal IoT networks that are not aware of oneM2M. This includes exposing non-oneM2M devices, applications and services to oneM2M entities via the oneM2M resource architecture, as well as exposing oneM2M functions and services to Proximal IoT networks that are not aware of oneM2M. This present document is independent of any specific Proximal IoT technology. Details for interworking with specific Proximal IoT Technologies are contained in other Technical Specifications.

Template Version: January 2017 (Do not modify)

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. NO oneM2M PARTNER TYPE 1 SHALL BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY THAT PARTNER FOR THIS DOCUMENT, WITH RESPECT TO ANY CLAIM, AND IN NO EVENT SHALL oneM2M BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. oneM2M EXPRESSLY ADVISES ANY AND ALL USE OF OR RELIANCE UPON THIS INFORMATION PROVIDED IN THIS DOCUMENT IS AT THE RISK OF THE USER.

Contents

1 Scope

The present document defines general guidelines when interworking between external Proximal IoT technologies which are not aware of oneM2M-defined functionality, and the oneM2M system (i.e. the interaction between non-oneM2M-aware devices, gateways or applications (non-oneM2M entities) and oneM2M-defined entities). In the present document guidelines are defined on how to use oneM2M-defined resources to represent the state, events, actions, procedures, services provided by the non-oneM2M entities and how to expose oneM2M functions or services represented by oneM2M-defined resource to non-oneM2M Proximal IoT technologies. Therefore, services provided by non-oneM2M entities can be consumed by oneM2M entities via the oneM2M defined interfaces and vice versa. When following these guidelines, oneM2M-aware entities consuming services provided by non-oneM2M-aware entities via the specified interworking methods do not need to know anything about external Proximal IoT technologies. Also entities in an external Proximal IoT network that are not oneM2M-aware can consume services provided by oneM2M entities when exposed to the external Proximal IoT network according to the specified methods.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

The following referenced documents are necessary for the application of the present document.

[1] oneM2M TS-0011: "Common Terminology".

[2] oneM2M TS-0001: "Functional Architecture".

[3] oneM2M TS-0023: "Home Appliances Information Model and Mapping".

[4] oneM2M TS-0022: "Field Device Configuration".

[5] oneM2M TS-0003: "Security Solutions".

[6] oneM2M TS-0034: "Semantics Support".

[7] oneM2M TS-0002: "Requirements".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] oneM2M Drafting Rules.

NOTE: Available at http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in oneM2M TS-0011 [1], oneM2M TS-0002 [7] and the following apply:

NOTE: A term defined in the present document takes precedence over the definition of the same term, if any, in oneM2M TS-0011 [1] and oneM2M TS-0001 [2].

proximal IoT: IoT components communicating with each other directly in a local network using specific communication protocols and information models

NOTE 1: The notion of "proximal" is motivated by the fact that many of such IoT technologies are based on discovery and advertisement techniques that are designed to be used primarily in networks where all communicating entities are in close proximity with each other, In the current context "proximal" does not imply spatial proximity rather than being part of the same IoT network that is in general not using any oneM2M-defined-functionality.

NOTE 2: When not stated otherwise, entities in a Proximal IoT network are not aware of any oneM2M-defined functions or procedures.

proximal IoT interworking: exchange of information and exposure/consumption of services across the borders between entities designed for non-oneM2M-defined Proximal IoT technologies and oneM2M entities

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in oneM2M TS-0011 [1] and the following apply:

IoT Internet of Things

4 Conventions

The key words "Shall", "Shall not", "May", "Need not", "Should", "Should not" in this document are to be interpreted as described in the oneM2M Drafting Rules [i.1]

5 Introduction

The scope of Proximal IoT Interworking is to enable the exchange of information between different things, devices and applications and the use of services they provide, irrespective of whether they are designed as oneM2M-defined entities according to the Functional architecture specified in oneM2M TS-0001 [2] or according to other non-oneM2M-defined Proximal IoT technologies. Proximal IoT Interworking can be modelled to be composed of actions on several layers: On the connection layer, on the resource framework layer and on the information model layer:

- Interworking on the connection layer - focus on the connection of entities. Two entities are interworkable if they support the same communication interface and communication protocol. Examples include Wifi connection, 3GPP wireless connection, etc. If two entities are interworkable on the connection layer, it is only guaranteed that data could be sent from one to another.

- Interworking on the resource framework layer - focus on the data types, resource template and data schemas. Two entities are interworkable if they share the same serializations, data types and resource templates. For example, if both entity can share information with the common understanding of xml schema, each entity will be able to recover the complete information contained in the message. Examples include SOAP, REST API, with specified serializations, etc.

- Interworking on the information model layer - focus on the information model, data model and common semantic understanding. Two entities are interworkable if they share the same information model and semantics. For example, in a smart home scenario, a light switch, a home gateway and an application that share the same information model can actually deploy the service of switching on and switching off the light if all of them use an information element with content "ON" to represent switching on the light and "OFF" to represent switching off the light. If the light switch is using "ON" but the application is using "TRUE" , the service cannot be deployed.

Interworking on the resource framework layer depends on the connection layer, and interworking on the information model layer depends on the resource framework layer.

To enable such consistent exchanges, oneM2M has designed the entire end to end architecture spanning entities for the platform (IN-CSE), gateways (MN-CSE) to devices (ASN and ADN), as described in oneM2M TS-0001 [2]. Corresponding to each layer, oneM2M has specified dedicated definitions for the enablement of:

- Interworking on the connection layer - Bindings defined by oneM2M i.e. HTTP, CoAP, MQTT and Websocket binding and associated procedures.
- Interworking on the resource framework layer - Serializations and resource structures defined by oneM2M.
- Interworking on the information model layer - The definition or the import of existing information models including the associated procedures in oneM2M. For example the HAIM in home domain in oneM2M TS-0023 [3] and all specializations of <mgmtObj> for device management in various Technical Specifications oneM2M TS-0001 [2], oneM2M TS-0022 [4].

The focus of the present document is the interworking on the information model layer and the implications on how to represent external Proximal IoT functions with means of resource instances in the oneM2M system.

However, the set of resource structures defined by oneM2M is very loosely coupled with the service of devices which may still cause interworking problems. Using CRUDN operations [2] on resources defined by oneM2M is the mechanism to enforce the common services oneM2M is trying to deliver. How to use these common services relies on interpretation of the implementer of the standard. For devices designed for non-oneM2M Proximal IoT technologies, if services of these devices are exposed to oneM2M entities using resources in inconsistent ways, it is still very hard to enable the interworking with these devices, because consumers of the services may need additional adaptation depending on different interpretations of resource content and relationships in different implementations.

In the present document, a general interworking architecture and framework to enable interworking up to the information model layer is defined.

6 General interworking architecture

6.1 Concept of Representation

In the oneM2M system, any kind of device, application or service or more generally speaking any kind of function that needs to be exposed to oneM2M-specified entities such as CSEs or AEs is represented by instances of specified resource types. Interaction with these termed functions is enabled by executing operations (e.g. create, delete, retrieve, update) on the resource instances. For example, by updating an attribute of a particular resource instance, an AE can change the state of a light actor from on to off. In addition, the concept of subscribing to resource instances and receiving notifications about content changes is also specified in oneM2M to allow for efficient monitoring of resource instances and thus the exposed function(s).

Following that fundamental concept of representing all functions by resource instances and their operations, real world devices or things in Proximal IoT networks that were not designed according to oneM2M specifications including the applications running on devices and/or services provided by devices can be represented by resource instances in the oneM2M system if interworking with those devices or things is needed. The resource instances are digital representations for the real world devices or things that are exposed to other entities in the oneM2M system via the oneM2M-specified interfaces for executing operations on those resource instances. How such resource instances, representing non-oneM2M Proximal IoT functions, are to be created and managed is described in the remainder of the present document. Furthermore, it is also described how to expose functions that are natively accessible in the oneM2M system, via the oneM2M-specified resource types and interfaces, to non-oneM2M Proximal IoT networks.

A representation of a non-oneM2M Proximal IoT function in a oneM2M-specified resource instance needs to be synchronized with the entity that it actually represents. For instance, if an attribute of a resource instance that represents the power state of an external Proximal IoT light is modified, the actual light state needs to be modified accordingly. The remainder of the present document describes how such synchronization is accomplished. Essentially any request targeting the representation of a non-oneM2M Proximal IoT function results in the corresponding interaction with the external device or thing in the Proximal IoT network. Similarly, any action in a non-oneM2M Proximal IoT function that is associated with a representation in the oneM2M system contained in a oneM2M resource instance shall result in an according operation on that resource instance.

Adhering to this concept of representation, oneM2M entities (AEs and CSEs) are enabled to request services or information provided by external non-oneM2M devices or things - termed "non-oneM2M Proximal IoT functions" - in the same way as interaction with native oneM2M devices or things - termed native oneM2M functions - is done.

6.2 Role of IPE(s)

6.2.1 Fit with the Functional Architecture

oneM2M is using an Interworking Proxy Entity (IPE) to handle the interworking between the oneM2M system and external non-oneM2M Proximal IoT functions residing on Non-oneM2M Device Nodes (NoDNs). The functional relationship between a NoDN and the CSE that is supposed to host the interworking functionality is defined in the following, see also Annex F of oneM2M TS-0001 [2].

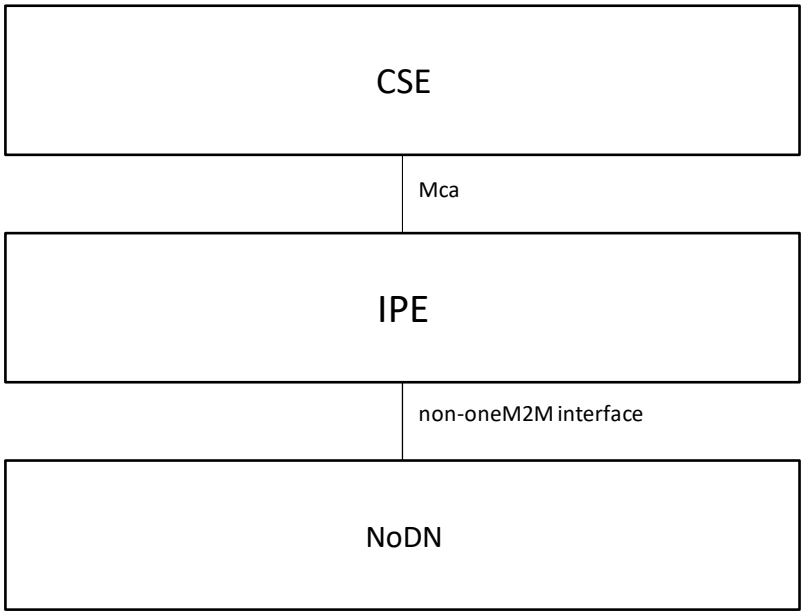

Figure 6.2.1-1: Interworking through IPE

An Interworking Proxy Entity is an AE that supports both, the oneM2M Mca reference point as well as the non-oneM2M interface that the functions on one or more connected NoDN(s) require. The IPE is responsible for synchronizing the services provided by the non-oneM2M Proximal IoT functions on the NoDN with oneM2M resource instances and vice versa. The IPE can be deployed together with the CSE or NoDN, or the IPE can also be deployed separately. Deployed together means running in the same execution environment or running on the same operating system where in such case, the communication between the deployed entities doesn't involve remote communication through wired or wireless network. In this kind of deployment, the communication method for supporting communication between the functional entities depicted in Figure 6.2.1-1 shall be provided by the execution environment or the operating system, e.g. inter-process communications, function calls, service call-backs or message bus technologies.

The IPE is registered to the CSE that is meant to host the interworking functionality and it translates the functions or services provided or consumed by one or more NoDN(s) to or from content of resource instances that are hosted by CSEs in the oneM2M system. When such resource instances represent functions or services provided by NoDN(s) connected to a specific IPE, the Registrar CSE of that IPE shall host those resource instances. The IPE shall translate any occurrence of operations on the oneM2M resource instances into invocation of the corresponding non-oneM2M Proximal IoT functions provided by the connected NoDN(s) and vice versa when non-oneM2M Proximal IoT functions executed on any connected NoDN(s) need to be reflected by change(s) of content in the corresponding representation(s) in oneM2M resource instances.

NOTE: More than one IPE may be instantiated in order to support interworking with an external non-oneM2M Proximal IoT technology. Also, an IPE may instantiate one or more supporting AEs that are used to simplify correlation with AE aspects such as service subscription profiles, access control privileges, authentication, authorization, etc.

6.2.2 Exposure of Proximal IoT functions to the oneM2M System

The role of an IPE, when it comes to exposure of external non-oneM2M Proximal IoT functions to the oneM2M System, includes the creation, monitoring, modification (update/delete) of resource instances that are supposed to represent those external functions on its own Registrar CSE. This role also includes the following:

- The IPE needs to determine which non-oneM2M Proximal IoT functions need to be exposed (e.g. through provisioning, discovery, on-demand signalling, etc.) and detect dynamic changes of the set of the non-oneM2M Proximal IoT functions to be exposed. On-demand discovery or change of exposure configurations may be triggered by other AEs/CSEs by modifying corresponding resource instances created by the IPE. A request to trigger discovery or to demand a change of the exposure configuration can be accomplished, for instance, via creation and monitoring of a <*container*> resource instance by the IPE, under which authorized AEs can create <*contentInstance*> resource instances, that indicate which action to take. Details of such a triggering mechanism are implementation depended and will not be further specified in the present document.

- The IPE needs to handle creation/deletion of resource instances representing non-oneM2M Proximal IoT functions according to the - possibly dynamically changing - need to expose them to the oneM2M system using resource types that are independent of the external Proximal IoT technology.

- The IPE is responsible to modify the resource instances representing the non-oneM2M Proximal IoT functions according to any state changes occurring in the external Proximal IoT system.

- The IPE is responsible for monitoring relevant changes in the resource instances representing the non-oneM2M Proximal IoT functions and invocation of appropriate non-oneM2M Proximal IoT function(s) when any operation(s) meant to trigger the execution of that non-oneM2M Proximal IoT function(s) occur for those resource instances in the hosting CSE.

The set of responsibilities of the IPE when exposing non-oneM2M Proximal IoT functions to the oneM2M system is summarized in Figure 6.2.2-1. The dashed boxes describe optional/alternative means to determine the set of exposed functions. Note that, in this Figure one IPE is responsible for all interworking actions. More than one IPE may be used to interwork with one particular Proximal IoT network. Also additional AEs may get instantiated by an IPE to support interworking, see clause 6.1. Details on the resource mapping are contained in clause 7.

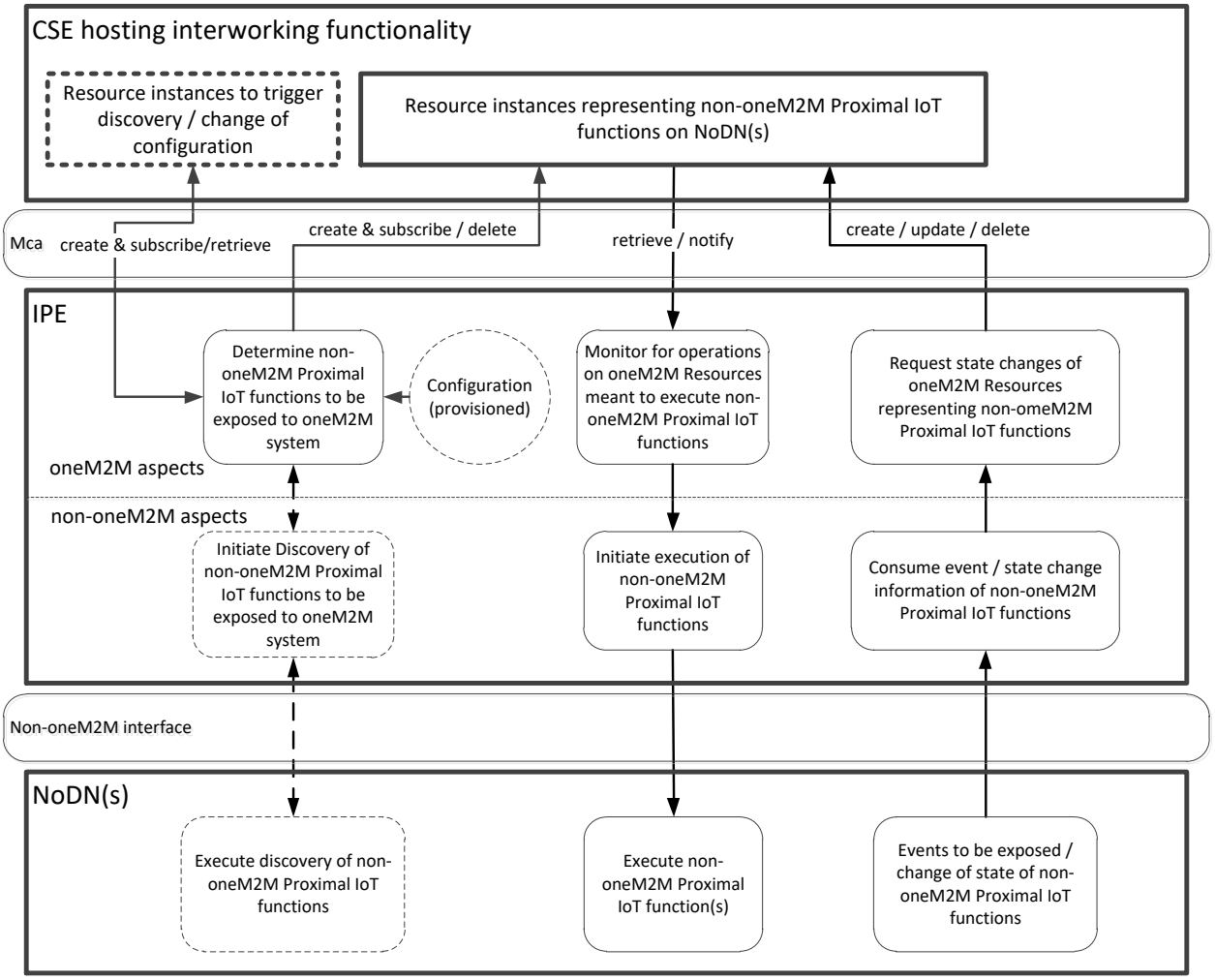

Figure 6.2.2-1: Exposure of Proximal IoT functions to the oneM2M System

6.2.3 Exposure of native oneM2M functions to the Proximal IoT System

The role of an IPE, when it comes to exposure of native oneM2M functions (aspects of devices, applications, services) to an external non-oneM2M Proximal IoT network, includes the monitoring, modification (update/delete) of resource instances that are representing oneM2M-internal functions to be exposed to an external non-oneM2M Proximal IoT network. This role also includes the following:

- The IPE needs to determine which native oneM2M functions need to be exposed to the external non-oneM2M Proximal IoT network (e.g. through provisioning, resource instance discovery, on-demand signalling, etc.) and detect dynamic changes of the set of native oneM2M functions to be exposed. On-demand discovery or change of exposure configurations may be triggered by other AEs/CSEs by modifying corresponding resource instances created by the IPE. A request to trigger discovery or to demand a change of the exposure configuration can be accomplished, for instance, via creation and monitoring of a <*container*> resource instance by the IPE, under which authorized AEs can create <*contentInstance*> resource instances, that indicate which action to take. Details of such a triggering mechanism are implementation depended and will not be further specified in the present document.

- The IPE needs to handle creation/deletion of functions in the non-oneM2M Proximal IoT network representing native oneM2M functions according to the - possibly dynamically changing - need to expose them to the non-oneM2M Proximal IoT network.

- The IPE is responsible to modify the resource instances representing exposed native oneM2M functions according to any events occurring in the external non-oneM2M Proximal IoT network that require state changes of the resource instances representing the exposed native oneM2M functions.

- The IPE is responsible for monitoring relevant changes in the resource instances representing exposed native oneM2M functions and invocation of the corresponding non-oneM2M Proximal IoT function(s) provided by the IPE when any operation(s) occur for those resource instances that require an indication of the changed state in the provided external functions.

The set of responsibilities when exposing native oneM2M functions to the external non-oneM2M Proximal IoT network is summarized in Figure 6.2.3-1. The dashed boxes or lines describe optional/alternative means to determine the set of exposed functions. Note that, in this Figure one IPE is responsible for all interworking actions. More than one IPE may be used to interwork with one particular Proximal IoT network. Also additional AEs may get instantiated by an IPE to support interworking, see clause 6.1. Details on the resource mapping are contained in clause 7.

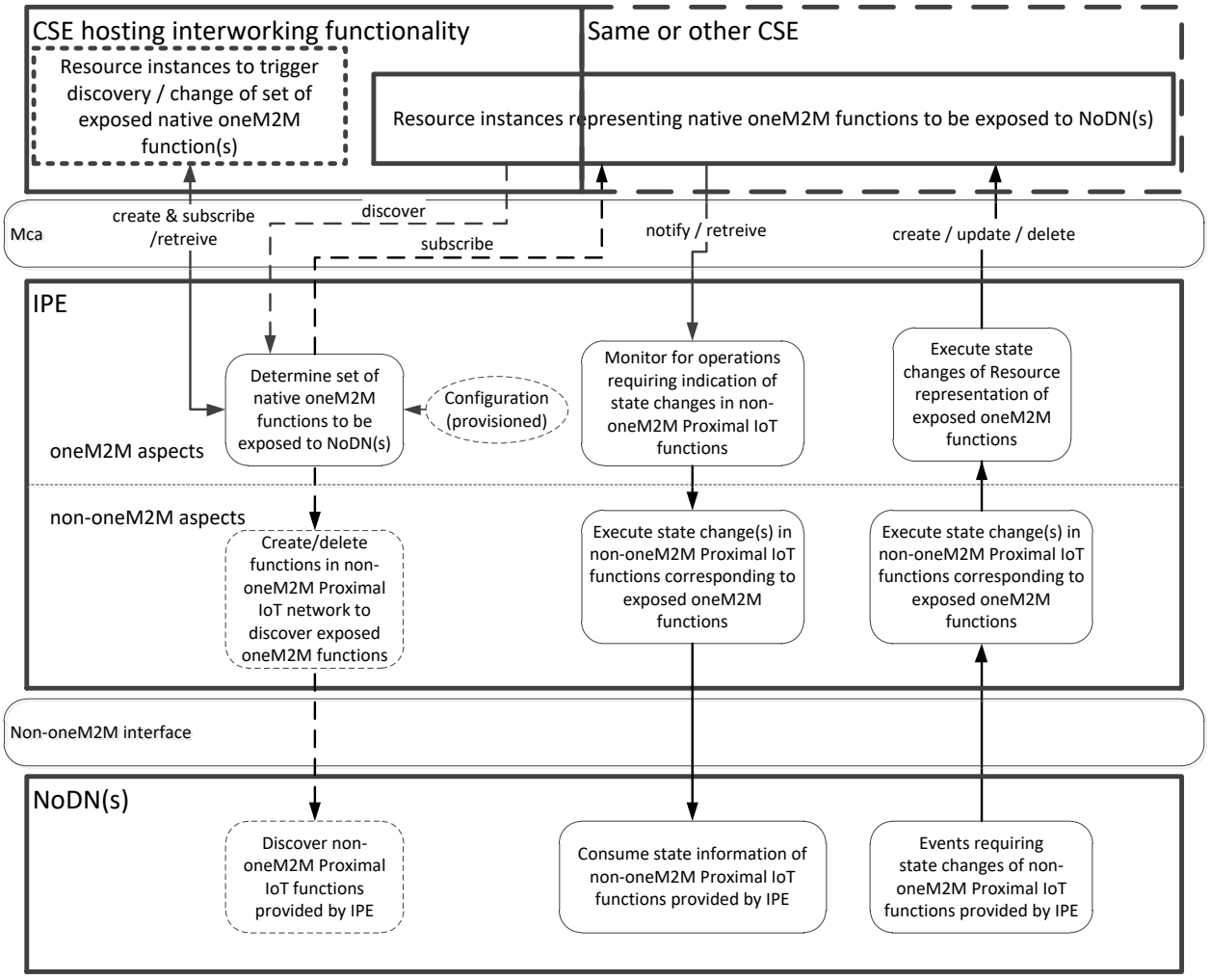

Figure 6.2.3-1: Exposure of native oneM2M functions to the Proximal IoT System

7 Representation of non-oneM2M entities in Proximal IoT networks

7.1 Representation of non-oneM2M Proximal IoT Devices

From a device management perspective in oneM2M, a device is represented using a <*node*> resource. All management related capabilities of a device are then represented using <*mgmtObj*> child resources of a <*node*> resource. This principle shall also be applied for non-oneM2M Proximal IoT devices (which are NoDNs), i.e. all aspects of device management of a device subject to device management methods defined in oneM2M should be exposed by using <*mgmtObj*> child resources of a <*node*> resource. The <*node*> resource instances representing device management aspects of non-oneM2M Proximal IoT devices shall be created by the responsible IPE on the IPE's Registrar CSE.

If the device complies to oneM2M-defined information models - such as the ones defined in oneM2M TS-0023 [3] - the device should be represented using the respective specializations of resources specified in oneM2M. For example, a home domain device for a light as defined in oneM2M TS-0023 [3] is represented using the corresponding specializations of <*flexContainer*> defined by the [*deviceLight*] resource type. If the information model of the device is not defined by oneM2M, a <*flexContainer*> may also be used with its *containerDefinition* attribute configured with a URI linking to the schema definition for that device type specified by the respective organization. Also if the <flexContainer> resource represents a non-oneM2M Proximal IoT device, the resource may be linked with the corresponding <*node*> resource that is used to reflect device management aspects of the device or to indicate relationship(s) to applications on the device represented by <*AE*> resource instances, if applicable. The instances of specializations of <*flexContainer*> resource types representing non-oneM2M Proximal IoT devices shall be created by the responsible IPE on the IPE's Registrar CSE. The preferred parent resource for such specialization of <*flexContainer*> resource instances is the IPE's own <*AE*> resource instance. The linkage between an instance of a specialization of the <*flexContainer*> resource type, representing a non-oneM2M Proximal IoT device, and the corresponding <*node*> resource instance, that is used to reflect device management aspects or relationships to applications of the device, shall be established as follows:

1) If present, a *nodeLink* attribute of the <*flexContainer*> specialization instance, representing the non-oneM2M Proximal IoT device, shall point to the <*node*> resource instance.

2) Otherwise, a *mgmtLink* attribute of the <*flexContainer*> specialization instance, representing the non-oneM2M Proximal IoT device, shall point to a <*deviceInfo*> resource instance that is a child of the <*node*> resource instance.

For devices that do not follow any standardized information model nor have any management requirements, there is no distinct resource types to be instantiated in the oneM2M system for the representation of the device.

7.2 Representation of non-oneM2M Proximal IoT Applications

A general pattern in oneM2M is the use of instances of an <*AE*> resource to represent applications running on devices/gateways if needed. In addition, all services provided by these applications should be represented as child resources of the respective <*AE*> resource instance. By browsing the child resources of an <*AE*> resource instance, it is easily understood what are all the services provided by the respective application. If an application de-registers (i.e. the <*AE*> resource instance is deleted) from the system, all the resources representing its services are also deleted since they are child resources of the <*AE*> instance. For example, if an application is to report temperature data, after registration, an <*AE*> resource instance representing the application is created on the registrar CSE. The data reporting service is then e.g. represented as a <*container*> or <*flexContainer*> child resource of the <*AE*> resource. If the <*AE*> resource gets deregistered, the <*container*> or <*flexContainer*> resource is deleted at the same time. The same principles should be applied to represent non-oneM2M Proximal IoT Applications on NoDN(s) and the services they provide (see clause 7.3).

According to the specific needs in a service deployment, the service provider may deploy one or multiple application instances on one device. Each NoDN application instance that needs to be exposed to the oneM2M system shall be represented by one <*AE*> resource instance and be assigned with one unique AE-ID to identify the application instance.

Depending on the type of oneM2M node hosting an application, its <*AE*> resource instance(s) may be hosted by different types of CSEs in the oneM2M system:

- For applications on ASNs and MNs, the corresponding <*AE*> resources shall be created under the <CSEBase> of the corresponding ASN-CSE and MN-CSE, accordingly.

- For applications on an ADN, the corresponding <*AE*> resources shall be created under the <CSEBase> of the Registrar CSE of the ADN (which may be an MN-CSE or IN-CSE).

If there is a need to represent applications on interworked NoDNs - which is the case if the interworked applications need to be identifiable for the purpose of service subscription, charging, differentiation during access control enforcement, authentication, App-ID registry, etc. - then one or more <*AE*> resource instances shall be created to represent those applications. The IPE is responsible to issue requests to the oneM2M system on behalf of the interworked applications by using the AE-ID of the created <*AE*> resources. Care should be taken for determining the number of necessary security associations for the created <*AE*> resources. If there is no need to represent different applications when interacting with functions on interworked NoDNs just one <*AE*> resource shall be created as the representation of the IPE that is responsible for accessing the NoDN services.

When a non-oneM2M Proximal IoT application running on a NoDN is represented by an <*AE*> resource instance and at the same time device management aspects or relationships to applications of that NoDN are represented by a <*node*> resource instance, the *nodeLink* attribute of that <*AE*> resource instance shall point to the <*node*> resource instance corresponding to that NoDN. Also the reverse linkage via the *hostedAELinks* attribute of the <*node*> resource shall be established.

7.3 Representation of non-oneM2M Proximal IoT Services

oneM2M defines different types of resources that may be used to represent services provided by a device. When representing non-oneM2M Proximal IoT services from interworked NoDN(s), proper resource types shall be chosen since the misusage of resource types for representing services may cause interoperability problems. General guidelines for resource representation of different services are as follows:

- For device management services: Specialized <*mgmtObj*> resource types as specified in oneM2M TS-0001 [2] and oneM2M TS-0022 [4], and <*mgmtCmd*>, <*execInstance*> as specified in oneM2M TS-0001 [2] shall be used. These resources shall be created by the responsible IPE as child resources of the <*node*> resource which represents the managed device (see clause 7.1).

- Home appliance services: Specialized <*flexContainer*> resource types for moduleClasses as specified in oneM2M TS-0023 [3] shall be used to represent those services.

- Data management services (not covered by oneM2M TS-0023 [3]): <*container*>, <*contentInstance*>, <*timeSeries*>, <*timeSeriesInstance*> as specified in oneM2M TS-0001 shall be used.

- Location services: <*locationPolicy*>, <*container*>, <*contentInstance*>, <*latest*>, <*oldest*> as specified in oneM2M TS-0001 [2] shall be used.

- Group services: <*group*>, <*fanOutPoint*>, <*localMulticastGroup*> as specified in oneM2M TS-0001 [2] shall be used

- Event/notification services: <*subscription*>, <*notificationTargetSelfReference*>, <*notificationTargetMgmtPolicyRef*>, <*notificationTargetPolicy*>, <*policyDeletionRules*> as specified in oneM2M TS-0001 [2] shall be used.

- Security services: <*accessControlPolicy*>, <*dynamicAuthorizationConsultation*>, <*role*>, <*token*>, <*authorizationDecision*>, <*authorizationPolicy*>, <*authorizationInformation*> as specified in oneM2M TS-0001 [2] and oneM2M TS-0003 [5] shall be used

- Semantic services: <*semanticDescriptor*>, <*ontologyRepository*>, <*ontology*>, <*semanticValidation*>, <*semanticMashupJobProfile*>, <*semanticMashupInstance*>, <*mashup*>, <*semanticMashupResult*> as specified in oneM2M TS-0001 [2] and oneM2M TS-0034 [6] shall be used.

- Charging services: <*statsConfig*>, <*eventConfig*>, <*statsCollect*> as specified in oneM2M TS-0001 [2] shall be used.

There are two ways of expressing relationships between resources as well as relationships between the services these resources represent: Parent-child relationship and linkage relationship. The linkage relationship only applies to specific oneM2M resource types such as <*accessControlPolicy*>, announced resources, and <*mgmtObj*> resources, etc.

The parent-child relationship of resources shall be used when the service represented by the child resource cannot exist independent of the services represented by the parent resource. If the parent service is deleted, the child services shall be deleted automatically.

History

Publication history		
V3.0.0	April 2019	Release 3 - Publication