ONEM2M TECHNICAL SPECIFICATION

Document Number	TS-0035-V3.0.0
Document Name:	OSGi Interworking
Date:	2019-04-24
Abstract:	The document defines principles and guidelines on interworking OSGi based devices and gateways to oneM2M system.

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

Contents

1 Scope 4

2 References 4

2.1 Normative references 4

2.2 Informative references 4

3 Abbreviations 4

4 Conventions 5

5 OSGi Interworking General Architecture 5

5.1 OSGi Interworking Architecture 5

6 Mapping of OSGi DAL 6

6.1 Introduction 6

6.2 Device service 6

6.3 Function service 7

6.3.1 Introduction 7

6.3.2 BooleanControl Function 8

6.3.3 BooleanSensor Function 8

6.3.4 MultiLevelControl Function 9

6.3.5 MultiLevelSensor Function 9

6.3.6 Meter Function 10

6.3.7 Alarm Function 10

6.3.8 Keypad Function 10

6.3.9 WakeUp Function 10

6.4 Device service procedure 10

6.5 Function service procedure 11

History 13

1 Scope

The present document defines principles and guidelines on how to interwork devices and gateways that comply to the OSGi framework to the oneM2M system. The interworking includes service exposure between an OSGi device or gateway and the oneM2M system. With the interworking, OSGi defined services can be made available by oneM2M defined resources. As a result, by making requests to oneM2M resources, applications can access the services provided by OSGi devices or gateways.

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-0023: "Home Appliances Information Model and Mapping".

[2] oneM2M TS-0033: "Proximal IoT Interworking".

NOTE: Available at http://www.onem2m.org/technical/published-drafts.

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.

[i.2] OSGi Residential.

NOTE: Available at https://osgi.org/download/r6/osgi.residential-6.0.0.pdf.

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

DAL	Device Abstraction Layer
AE	Application Entity
CSE	Common Services Entity
SDT	Smart Device Template
DMT	Device Management Tree
IPE	Interworking Proxy Entity
HAIM	Home Appliance Information Model
UID	Unique Identifier
UIDS	Unique Identifiers
TS	Technical Specification

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 OSGi Interworking General Architecture

5.1 OSGi Interworking Architecture

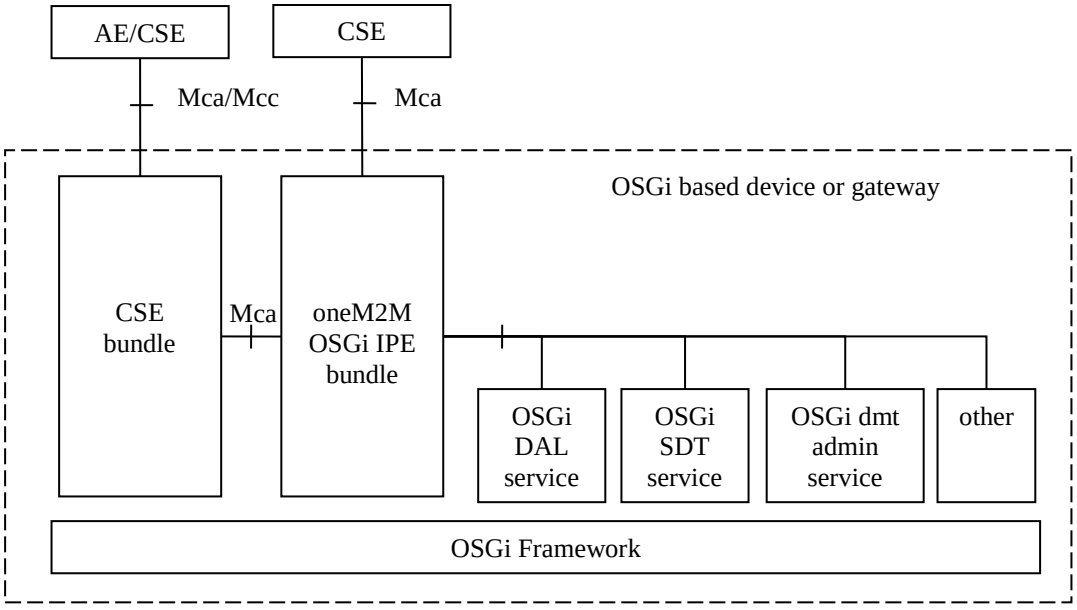

Figure 5-1: OSGi Interworking Architecture

OSGi Interworking is to interwork services provided by an OSGi based device or gateway to oneM2M entities which may be an AE or CSE. The OSGi services may include an OSGi defined DAL (Device Abstraction Layer) service, SDT (Smart Device Template) service, DMT (device management tree) admin service, other standardized services or proprietary services. The oneM2M-OSGi IPE bundle is in charge of the interworking of OSGi services to oneM2M resources and vice versa.

The IPE bundle maps the invocation of the OSGi services and changes of state of oneM2M resources. If the OSGi based device or gateway hosts a CSE bundle, the IPE interacts with the CSE bundle internally. The OSGi based device or gateway interacts with the other oneM2M entities through Mca or Mcc reference point. If the OSGi based device or gateway does not host a CSE bundle, the IPE interacts with the CSE through a network interface.

The principle of how the interworking should be done shall follow the definition in oneM2M TS-0033 [2] proximal IoT interworking. For devices described by HAIM (Home Appliance Information Model) in oneM2M TS-0023 [1], <*flexContainer*> resources shall be used to represent the OSGi services. oneM2M also allows the transparent interworking of OSGi services through <*container*> resources, <*AE*> resources and <*node*> resources as defined in oneM2M TS-0033 [2].

6 Mapping of OSGi DAL

6.1 Introduction

The OSGi DAL (Device Abstraction Layer) [i.2] is an OSGi defined service to unify interfaces for accessing devices. By using the OSGi DAL, application developers do not need to deal with protocol details to interact with different types of devices such as sensors, actuators, etc.

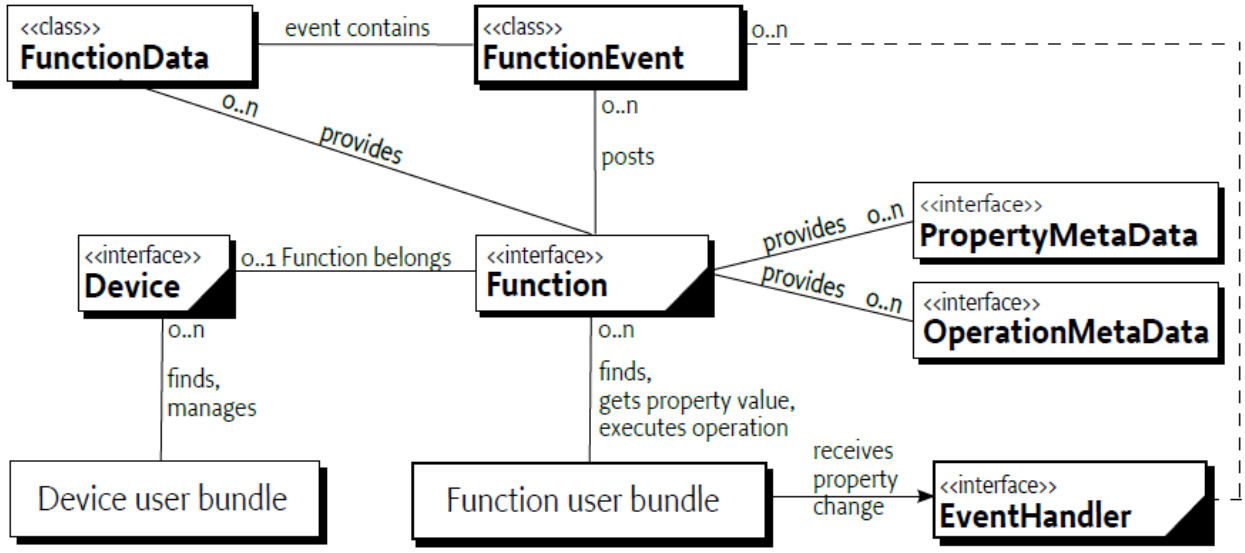

Figure 6.1-1: OSGi DAL [i.2]

OSGi DAL is comprised of several services:

- Device: represents the registered device in the OSGi framework. The device service contains properties describing the device's metadata and context information. The device service has one or more function services.
- Function: represents the function provided by the device. The function could be data reporting or actuating. The function provides a set of FunctionData, properties and operations. A property is the data information that can be accessed by the application. An operation is the interface that can be invoked by an application to trigger a certain procedure. The function also posts FunctionEvent.
- FunctionEvent: represents the asynchronous event of the function. Once the event is triggered, the event is handled by the registered event handler.
- FunctionData: represents the data structure that contains the property and metadata of the function.
- PropertyMetadata: represents the metadata of the function property.
- OperationMetadata: represents the metadata of the function operation.

6.2 Device service

The device service defined by OSGi DAL maps to a oneM2M NoDN and is represented as a specialized <*flexContainer*> resource and a <*node*> resource.

The OSGi device service itself shall be mapped to a specialized <*flexContainer*> resource of a specific device model that the OSGi device service instance represents. The properties of an OSGi device service shall be mapped to the attributes of the <*flexContainer*> resource, the <*node*> resource, and its child *[deviceInfo]* resource.

Once an OSGi device service is registered at the OSGi framework, the IPE shall be responsible for acquiring all OSGi device service properties and other related services (such as OSGi function services) and creating the corresponding resources on the oneM2M CSE.

Upon the registration of an OSGi device service, the IPE should create a <*node*> resource and a *[deviceInfo]* specialization as the child resource of the <*node*> resource. The IPE should also create one <*flexContainer*> resource with specialized mandatory *[customAttribute]* as a *'nodeLink'* attribute, which links to a <*node*> resource that is hosted on the same hosting CSE of the <*flexContainer*>. The mapping between OSGi device and oneM2M resources shall be as follows.

Table 6.2-1: Mapping of OSGi device service to oneM2M resources

OSGi	oneM2M	Description
OSGi device service	<*node*> resource, *[deviceInfo]* resource, <*flexContainer*> resource	The <*node*>, *[deviceInfo]* and <*flexContainer*> resources are created upon the available of OSGi device service.
SERVICE_UID property	See description	Maps to the resourceID of the <*flexContainer*> allocated by the Hosting CSE, however the value does not need to be same. The mapping between SERVICE_UID and resourceID is maintained by IPE internally.
SERVICE_REFERENCE_UIDS property	See description	Reference device maps to the sub-device defined in oneM2M. Therefore, the reference device in OSGi is mapped to the hierarchical relationship of <*flexContainer*> resources.
SERVICE_DRIVER property	*areaNwkType* attribute of *[areaNwkInfo]* resource	Service driver maps to the area network type.
SERVICE_NAME property	*resourceName* attribute of <*flexContainer*> resource	The SERVICE_NAME is used to request resource name of <*flexContainer*> resource.
SERVICE_STATUS property	See description	Is maintained by lifecycle management of <*flexContainer*> resources. The <*flexContainer*> resource should only be created if the SERVICE_STATUS of the OSGi device is STATUS_ONLINE.
SERVICE_STATUS_DETAIL property	See description	The status is not visible to oneM2M. The IPE will monitor the status change and reflect the status change in the lifecycle management of the <*flexContainer*> resource.
SERVICE_HARDWARE_VENDOR property	*manufacturer* attribute of *[deviceInfo]* resource	-
SERVICE_HARDWARE_VERSION property	*hwVersion* attribute *[deviceInfo]* resource	-
SERVICE_FIRMWARE_VENDOR property	*labels* attribute of *[deviceInfo]* resource	-
SERVICE_FIRMWARE_VERSION property	*fwVersion* attribute of *[deviceInfo]* resource	-
SERVICE_TYPES property	*deviceType* attribute of *[deviceInfo]* resource	-
SERVICE_MODEL property	*model* attribute of *[deviceInfo]* resource	-
SERVICE_SERIAL_NUMBER property	*deviceLabel* attribute of *[deviceInfo]* resource	-

6.3 Function service

6.3.1 Introduction

An OSGi function service maps to a specialized <*flexContainer*> resource that correspond to a moduleClass specified in oneM2M TS-0023 [1]. An OSGi function service may be mapped to different <*flexContainer*>s that correspond to different moduleClasses depending on the OSGi device service that the OSGi function service belongs to.

Table 6.3.1-1: Mapping of OSGi function to oneM2M resources

OSGi	oneM2M	Description
OSGi function service	moduleClass <*flexContainer*> resource	The function service is mapped to moduleClass <*flexContainer*>.
SERVICE_UID property	See description	Maps to resourceID of <*flexContainer*> resource. The resourceID is allocated by Hosting CSE. The mapping relationship is maintained by IPE internally.
SERVICE_TYPE property	See description	Maps to containerDefinition of <*flexContainer*> resource and is maintained by IPE.
SERVICE_VERSION property	*labels* attribute of the <*flexContainer*> resource	Versioning information of the OSGi service is mapped to *labels* attribute of the <*flexContainer*> resource.
SERVICE_DEVICE_UID property	See description	Maintained by the parent-child relationship of device model <*flexContainer*> and moduleClass <*flexContainer*>
SERVICE_REFERENCE_ UIDS property	*labels* attribute of the <*flexContainer*> resource	Mappes to the *labels* attribute of the <*flexContainer*> resource.
SERVICE_DESCRIPTION property	*labels* attribute of <*flexContainer*> resource	The description of the service maps to the labels attribute of <*flexContainer*> resource.
SERVICE_OPERATION_N AMES property	<*flexContainer*> resource for action	
SERVICE_PROPERTY_N AMES property	<*flexContainer*> resource for property	

6.3.2 BooleanControl Function

BooleanControl function is used for switch-type of device functions like a light, door, window or power socket. It maps to the binarySwitch <*flexContainer*> that is a moduleClass.

Table 6.3.2-1: Mapping of BooleanControl function to binarySwitch moduleClass

OSGi	oneM2M	Description
inverse	*toggle*	Toggle the switch.
setTrue	Update the *powerState* to true	
setFalse	Update the *powerState* to false	
data	*powerState*	The current state of the switch

6.3.3 BooleanSensor Function

BooleanSensor function is used to report the data of a device function such as a light, door, window or power socket. It maps to several different <*flexContainer*> resources that are moduleClass. It is based on the device type which determines the specialization of <*flexContainer*> to be mapped to.

Table 6.3.3-1: Mapping of BooleanSensor function to binarySwitch moduleClass

OSGi	oneM2M	Description
data	*powerState*	

Table 6.3.3-2: Mapping of BooleanSensor function to doorLock moduleClass

OSGi	oneM2M	Description
data	*doorLock*	

Table 6.3.3-3: Mapping of BooleanSensor function to boiler moduleClass

OSGi	oneM2M	Description
data	*status*	

Table 6.3.3-4: Mapping of BooleanSensor function to waterSensor moduleClass

OSGi	oneM2M	Description
data	*alarm*	

6.3.4 MultiLevelControl Function

The MultiLevelControl Function maps to different moduleClass depending on the device type.

Table 6.3.4-1: Mapping of MultiLevelControl function to brightness moduleClass

OSGi	oneM2M	Description
data	*brightness*	

Table 6.3.4-2: Mapping of MultiLevelControl function to foaming moduleClass

OSGi	oneM2M	Description
data	*foamingStrength*	

6.3.5 MultiLevelSensor Function

The MultiLevelSensor Function maps to different moduleClass depending on the device type.

Table 6.3.5-1: Mapping of MultiLevelControl function to height moduleClass

OSGi	oneM2M	Description
data	*height*	

Table 6.3.5-2: Mapping of MultiLevelControl function to weight moduleClass

OSGi	oneM2M	Description
data	*weight*	

Table 6.3.5-3: Mapping of MultiLevelControl function to liquidRemaining moduleClass

OSGi	oneM2M	Description
data	*liquidRemaining*	

Table 6.3.5-4: Mapping of MultiLevelControl function to brightness moduleClass

OSGi	oneM2M	Description
data	*brightness*	

Table 6.3.5-5: Mapping of MultiLevelControl function to foaming moduleClass

OSGi	oneM2M	Description
data	*foamingStrength*	

6.3.6 Meter Function

The Meter Function maps to different moduleClass depending on the device type.

Table 6.3.6-1: Mapping of Meter function to energyConsuption moduleClass

OSGi	oneM2M	Description
current	*roundingEnergyConsumption*	
total	*absoluteEnergyConsumption*	

6.3.7 Alarm Function

The Alarm Function maps to different moduleClass depending on the device type.

Table 6.3.7-1: Mapping of Alarm function to motionSensor moduleClass

OSGi	oneM2M	Description
alarm	*alarm*	

Table 6.3.7-2: Mapping of Alarm function to smokeSensor moduleClass

OSGi	oneM2M	Description
alarm	*alarm*	

Table 6.3.7-3: Mapping of Alarm function to temperatureAlarm moduleClass

OSGi	oneM2M	Description
alarm	*alarm*	

Table 6.3.7-4: Mapping of Alarm function to waterSensor moduleClass

OSGi	oneM2M	Description
alarm	*alarm*	

6.3.8 Keypad Function

The Keypad Function maps to keypad moduleClass of oneM2M.

Table 6.3.8-1: Mapping of Keypad function to keypad moduleClass

OSGi	oneM2M	Description
key	*keyNumber*	

6.3.9 WakeUp Function

This Function currently has no corresponding moduleClass in oneM2M.

6.4 Device service procedure

The IPE is responsible for monitoring the OSGi Device service and synchronizing the properties of the Device service to attributes of the oneM2M resources.

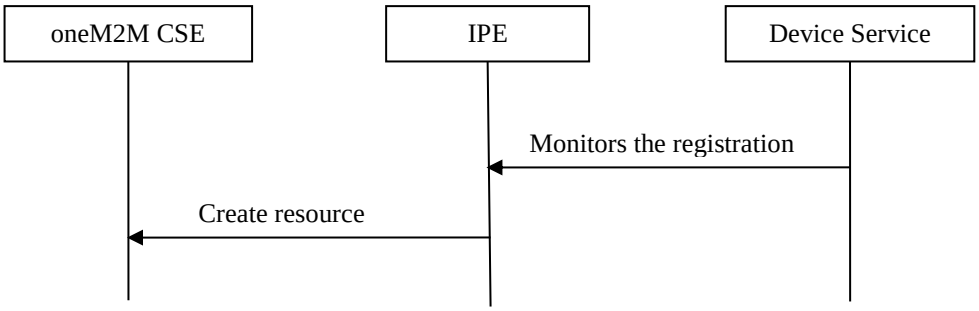

Figure 6.4-1: Procedure of registering Device service

The mapping of resources shall follow the definition in oneM2M TS-0033 [2] proximal IoT interworking. Depending on the available properties of the Device service, <*node*> resource, <*AE*> resource, <*container*> resource or <*flexContainer*> resource may be used to represent the Device service.

6.5 Function service procedure

The IPE is responsible for monitoring the Function property and synchronizing the change of the property to oneM2M resources. The IPE is also responsible for monitoring the update of the oneM2M resource that corresponds to the Function operation. Upon receiving the request targeting the resource, the IPE shall invoke the corresponding Function operation.

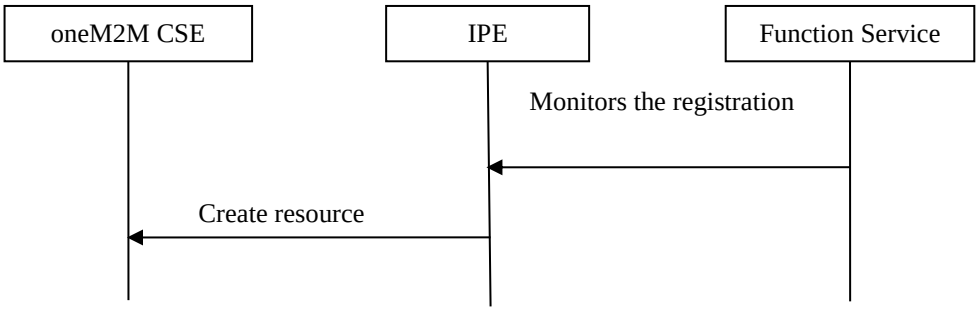

Figure 6.5-1: Procedure of registering function service

Monitors the registration: The IPE monitors the registration of the Function service to the OSGi framework.

Create resource: The IPE creates the corresponding resource to the oneM2M CSE.

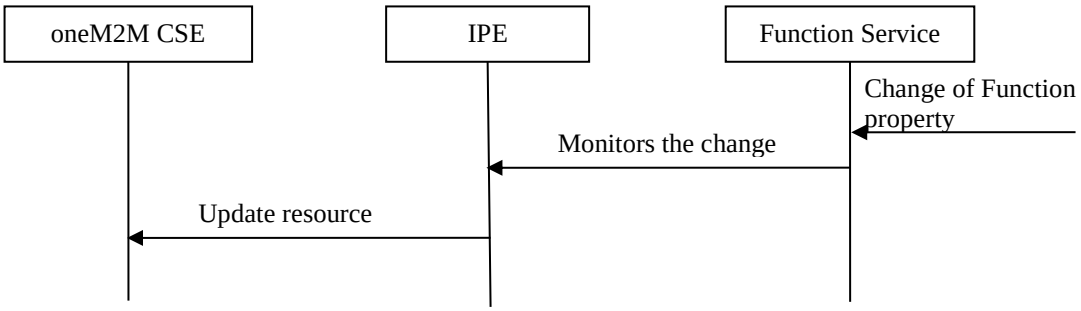

Figure 6.5-2: Procedure of changing function property

Change of Function property: The Function property may be changed due to various reasons such as hardware triggered break, sensor change, local application change etc.

Monitors the change: The IPE monitors the change by subscribing to the eventable properties, acquiring the property periodically or by some other internal call back functions.

Update the resource: The IPE updates the corresponding resource of the Function.

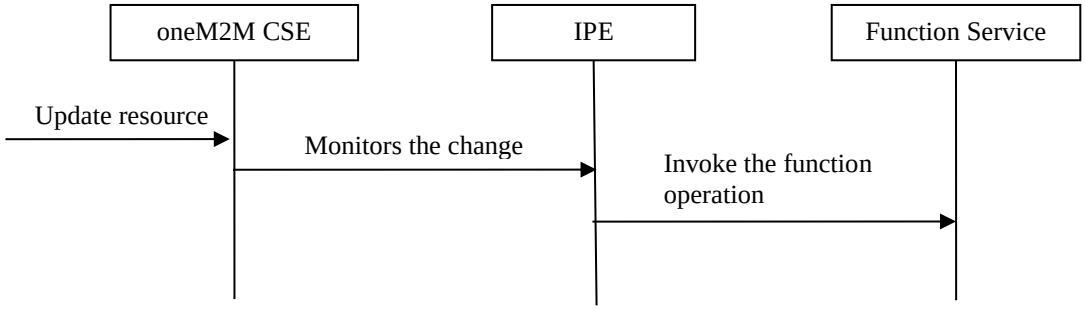

Figure 6.5-3: Procedure of invoking function operation

Update resource: The oneM2M CSE receives update requests from applications to update the resource that corresponds with the Function service.

Monitors the change: The IPE monitors the change by subscribing to the resource and receiving notifications or polling etc.

Invoke the function operation: The IPE invokes the function operation provided by the Function service.

History

Publication history		
V3.0.0	April 2019	Release 3 - Publication