oneM2M
Technical Report

Document Number	TR-0039-V-2.0.0
Document Name:	Developer guide: Interworking Proxy using SDT
Date:	2018-03-12
Abstract:	The document describes how a developer can easily implement interworking with non-oneM2M devices using the SDT (Smart Device Template) defined in TS-0023 (Home appliances Information Model).
Template Version: January 2017 (Do not modify)	

Contents

1 Scope

The present document describes how a developer can quickly and easily implement interworking with non-oneM2M devices using the SDT (Smart Device Template) defined in TS-0023 [i.2] (Home appliances Information Model). As an example, a scenario of an application controlling and monitoring a connected lamp, already commercially available on the market, is proposed. The goal is to describe with an example the methodology for building this interworking, allowing the developers to apply it to any other non-oneM2M devices using the abstraction layer provided by SDT.

To focus on the topics described above, the security aspect is not considered in the scope of this developer guide, especially Access Control Policy issue is not discussed

2 References

2.1 Normative references

Normative references are not applicable in the present document.

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

[i.3] oneM2M TS-0001: "Functional Architecture".

[i.4] oneM2M TS-0004: "Service Layer Core protocol Specification".

[i.5] oneM2M TS-0009: "HTTP Protocol Binding".

3 Abbreviations

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

ADN-AE	AE which resides in the Application Dedicated Node
AE	Application Entity
CSE	Common Services Entity
CSE-ID	Common Service Entity Identifier
IN	Infrastructure Node
IN-AE	Application Entity that is registered with the CSE in the Infrastructure Node
IN-CSE	CSE which resides in the Infrastructure Node
MN-AE	Application Entity that is registered with the CSE in Middle Node
MN-CSE	CSE which resides in the Middle Node
URI	Uniform Resource Identifier

4 Conventions

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

5 Use Case

5.0 Introduction

This clause describes a use case which helps to understand the devices abstraction layer concept of oneM2M through the combined use of the SDT-based (Smart Device Template) Information Model specified in oneM2M TS-0023 [i.2] and the Interworking Proxy Entity.

5.1 Abstraction description via the lightbulbs example

This guide is based on the following use case: an application on a Gateway, Cloud/Server, or Smartphone wants to monitor and control lightbulbs. Due to the abstraction layer provided by oneM2M, the monitoring and controlling processes can be performed independently from underlying connectivity technologies. This tutorial is based on the example of Philips Hue lightbulbs (these lightbulbs require dedicated bridge for connection with Gateway). Figure 5.1-1 presents the general architecture of the use case.

NOTE: The terms of use of Philips Hue API are available at https://developers.meethue.com/documentation/terms-use.

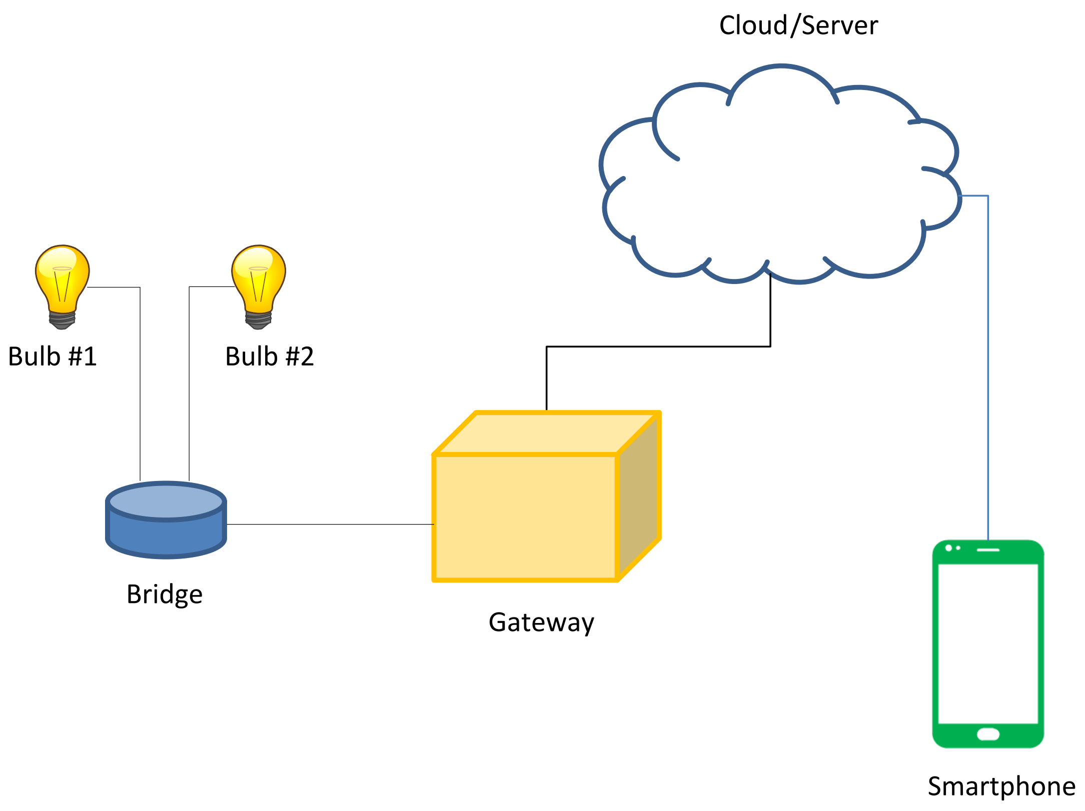


Figure 5.1-1: Overview of the lightbulbs use case

The main components are introduced as follows:

- The lightbulbs are deployed in a home and are attached to a home gateway via Philips Hue Bridge.

- The Gateway communicates with a Cloud/Server platform allowing the lightbulbs to be controlled remotely by the Smartphone.
- The Cloud/Server platform and Gateway supports a set of services to enable the Smartphone to more easily control and monitor the lightbulbs in the home. Some examples of services include device abstraction layer, registration, discovery, data management, group management, subscription/notification, etc.
- The Smartphone hosts an application used to remotely control and monitor the lightbulbs in the home and supports the following capabilities:
  - Discovery of the lightbulbs deployed in the home.
  - Sending commands to change light state i.e. switch on/off and change colour.
  - Receive state of a lightbulb.

6 Introduction to IPE and SDT

6.1 Introduction to IPE

An IPE (Interworking Proxy Entity) is a specialized AE (Application Entity) that allows the oneM2M system to interact with any non-oneM2M system, in a seamless way, through the Mca interface. It has the capability to remap the specific data model to oneM2M resources (<AE>, <container>, <flexContainer>, etc.) and maintain bidirectional communication with the non-oneM2M system.

6.2 Introduction to SDT

6.2.1 SDT data model description

The SDT (Smart Device Template) is a reference template to model most home appliance functions in a unified way which is a result of consensus amongst various SDOs and industry alliances. Abstraction from the various underlying home-area network technologies and getting an unified way of controlling/commanding the appliances are among the key goals of the SDT.

The SDT approach is to define re-usable basic functions (or services) (labelled "ModuleClass" in Figure 6.2-1) which can represent the typical functions found, for example, in many home automation systems, such as "on/off", "dim a lamp", "receive events from binary sensor", "read data from sensor", etc.

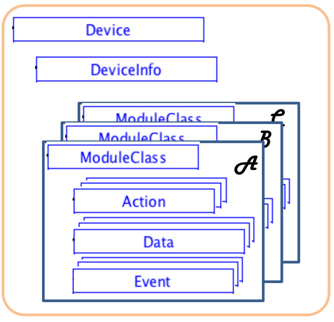


Figure 6.2.1-1: SmartHome Device Template for a generic device

The SDT supports the use of a set of templates for generic devices or appliances (e.g. for a dimmable lamp, a basic washing machine, etc., which would be specific instances of the "Device" object) which form the basis of APIs used by application developers. These templates can also be referenced by manufacturers creating XML documents to describe their specific products. For example, the SDT enables specification of a generic washing machine template, with on/off, set-wash-temperature, pause and a few other commands, which could be referenced by a manufacturer as the schema for a XML description of a basic model washing machine. The SDT allows for vendor-specific additional commands (ModuleClasses) to suit specific product types.

The SDT is available under Apache License 2 at oneM2M's GitLab: https://git.onem2m.org/MAS/SDT

6.2.2 SDT Device example: deviceLight

A light is a device that is used to control the state of an illumination device. This Device has one mandatory binarySwitch Module and the following optional Modules: faultDetection, runState, colour, colourSaturation, brightness.

Table 6.2.2-1: Modules of deviceLight Device model (from [1])

Module Instance Name	Module Class Name	Optional	Description
faultDetection	faultDetection	true	See clause 5.3.16
binarySwitch	binarySwitch	false	See clause 5.3.5
runState	runState	true	See clause 5.3.28
colour	colour	true	See clause 5.3.10
colourSaturation	colourSaturation	true	See clause 5.3.11
brightness	brightness	true	See clause 5.3.8

The faultDetection ModuleClass provides information about whether a fault has occurred in the actual device.

Table 6.2.2-2: DataPoints of faultDetection ModuleClass (from [1])

Name	Type	Readable	Writable	Optional	Documentation
status	xs:boolean	true	false	false	Status of fault detection.
code	xs:integer	true	false	true	Code of the fault.
description	xs:string	true	false	true	Message of the fault.

The binarySwitch ModuleClass provides capabilities to control and monitor the state of power.

Table 6.2.2-3: Actions of binarySwitch ModuleClass (from [1])

Return Type	Name	Argument	Optional	Documentation
none	toggle	none	true	Toggle the switch.

Table 6.2.2-4: DataPoints of binarySwitch ModuleClass (from [1])

Name	Type	Readable	Writable	Optional	Documentation
powerState	xs:boolean	true	true	false	The current status of the binarySwitch. "True" indicates turned-on, and "False" indicates turned-off.

The runState ModuleClass provides capabilities to control and monitor machine state of appliances.

Table 6.2.2-5: DataPoints of runState ModuleClass (from [1])

Name	Type	Readable	Writable	Optional	Documentation
currentMachineState	hd:machineState	true	true	false	Currently active machine state.
machineStates	list of hd:machineState	true	false	false	List of possible machine states the device supports (see clause 5.5.15)
progressPercentage	float	true	false	true	Indication of current progress in percentage

The colour ModuleClass provides the capabilities to set the value of Red, Green, Blue for the colour device.

Table 6.2.2-6: DataPoints of colour ModuleClass (from [1])

Name	Type	Readable	Writable	Optional	Documentation
red	xs:integer	true	true	false	The R value of RGB; the range is [0,255]
green	xs:integer	true	true	false	The G value of RGB; the range is [0,255]
blue	xs:integer	true	true	false	The B value of RGB; the range is [0,255]

The colourSaturation ModuleClass describes a colour saturation value. The value is an integer. A colourSaturation has a range of [0,100]. A colourSaturation value of 0 means producing black and white images. A colourSaturation value of 50 means producing device specific normal colour images. A colourSaturation value of 100 means producing device very colourfull images.

Table 6.2.2-7: DataPoints of colourSaturation ModuleClass (from [1])

Name	Type	Readable	Writable	Optional	Documentation
colourSaturation	xs:integer	true	true	false	The status of colour saturation level.

The brightness ModuleClass describes the brightness of a light e.g. from a lamp. Brightness is scaled as a percentage. A lamp or a monitor can be adjusted to a level of light between very dim (0 % is the minimum brightness) and very bright (100 % is the maximum brightness).

Table 6.2.2-8: DataPoints of brightness ModuleClass (from [1])

Name	Type	Readable	Writable	Optional	Documentation
brightness	xs:integer	true	true	false	The status of brightness level in percentage.

6.3 IPE and SDT in oneM2M tree

In reference to oneM2M terminology, when a new device is discovered the CSE should register it and map its SDT-based representation to oneM2M common resources according to the rules defined in TS-0023 [i.2].

To allow current non-oneM2M devices ("NoDN", Non-oneM2M Device Node) to connect to the oneM2M system the specification provides "Interworking/Integration of non-oneM2M solutions and protocols" section (Annex F of oneM2M TS-0023 [i.2]). Figure 6.3-1 shows SDT in oneM2M architecture.

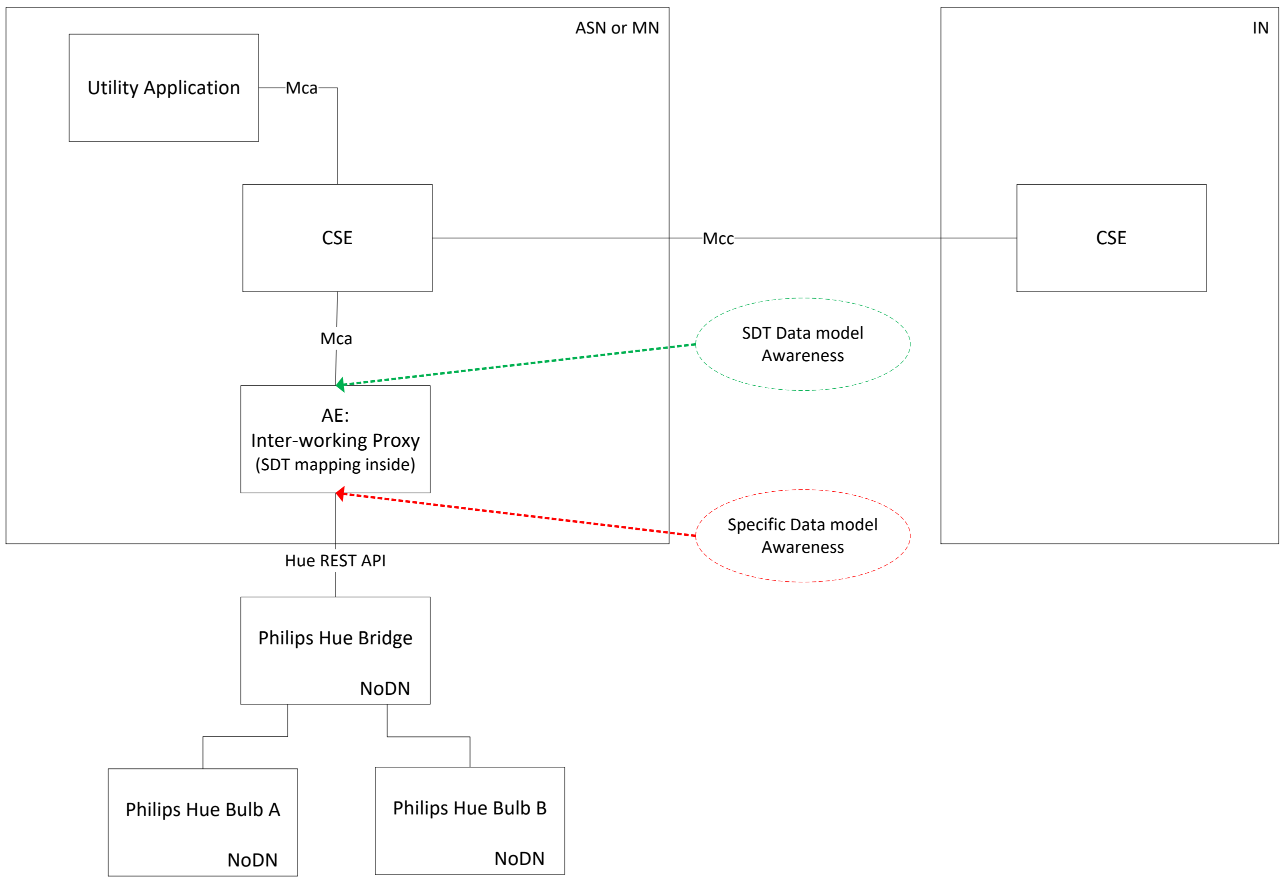


Figure 6.3-1: Translation of non-oneM2M Data Model to SDT Data Model via the IPE

7 Functional architecture

Clause 7 describes how the elements of this use case are represented by corresponding oneM2M architectural entities. Figure 7-1 presents the functional architecture of this use case.

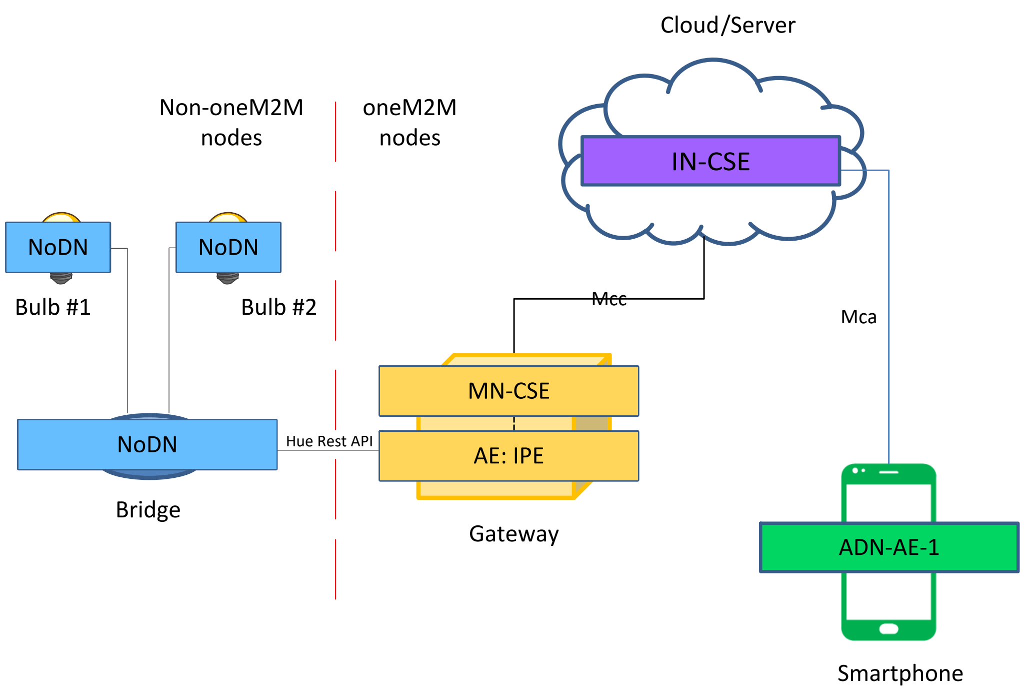


Figure 7-1: oneM2M functional architecture of the bulbs use case

There are two sections in the functional architecture of this use case. On the left side of the red dashed line there is the non-oneM2M section based on Philips Hue architecture. It consists of two Hue Bulbs and Hue Bridge.

On the right side there is the oneM2M architecture section which consists of MN-CSE (Middle Node - Common Service Entity) and AE: IPE (Application Entity: Interworking Proxy Entity) on the Gateway, IN-CSE (Infrastructure Node CSE) on the Cloud/Server, ADN-AE-1 (Application Dedicated Node) on the Smartphone and Mcc and Mca reference points.

Mca reference point is used to interface an AE and a CSE together. Mcc reference point is used to interface CSEs. In this use case Mca is used between ADN-AE-1 on the Smartphone and IN-CSE on the Cloud/Server, while Mcc is used between MN-CSE on the Gateway and IN-CSE on the Cloud/Server.

AE: IPE makes non-oneM2M devices visible to oneM2M platform, so it allows ADN-AE-1 on the Smartphone to remotely control and monitor the bulbs. AE: IPE mechanism is described in details in clause 6.3.

8 Procedures and call flows

8.1 Introduction

The deployment of the oneM2M standard in the present use case requires procedures that are classified as follows:

- **Registration:** AE registration with CSE: IPE-AE with MN-CSE, ADN-AE with IN-CSE.
- **Resources creation:** a set of *<flexContainer>* resources creation to represent SDT Device in a resource tree.
- **Discovery of resources with filter criteria:** discovery of resources with given containerDefinition attribute filtered with filter criteria to receive associated URI.
- **Discovery of children resources:** discovery of a resource tree associated with particular SDT Device to get access to particular ModuleClass, DataPoint, Action.
- **Get resource**: get value of DataPoints through sending RETRIEVE request
- **Change resource:** changing a DataPoint value or triggering an Action through sending UPDATE request
- **Subscription and notification mechanism**: monitor DataPoints values changes through subscription and notification mechanism

In this developer guide there is an assumption that MN-CSE is registered with IN-CSE. More assumptions are listed in clause 9.2.

8.2 Call Flows from perspective of device adapter developer

8.2.1 IPE-AE registration with MN-CSE

The procedure to register device adapter (IPE-AE) with gateway (MN-CSE) is as follows:

- Device adapter (IPE-AE) sends registeration request to Gateway (MN-CSE)
- Gateway (MN-CSE) responds with status of the registration and Content-Location header for the registered entity

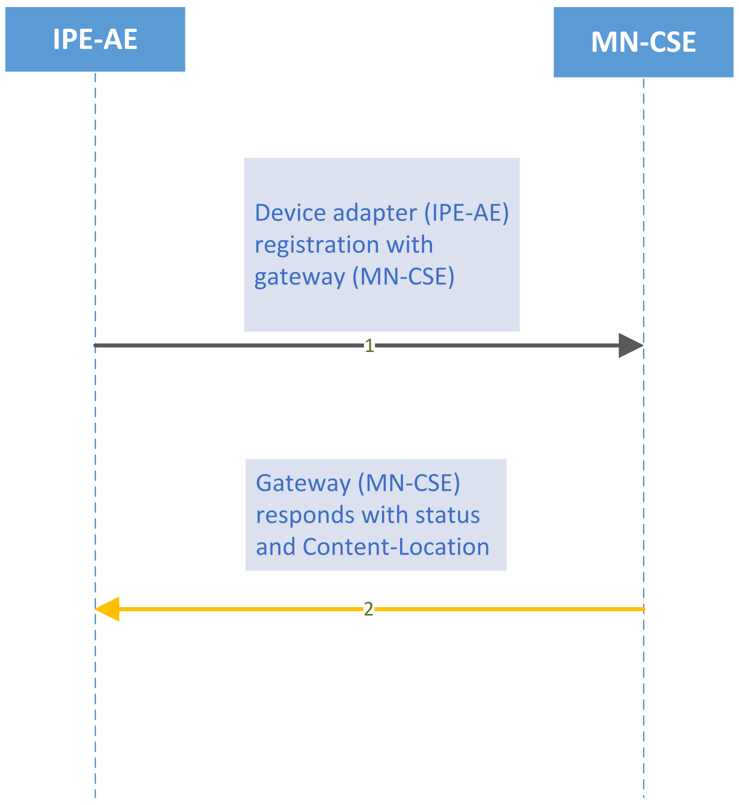

IPE-AE
MN-CSE
Device adapter (IPE-AE) registration with gateway (MN-CSE)
1
Gateway (MN-CSE) responds with status and Content-Location
2

Figure 8.2.1-1: AE registration with CSE request

8.2.2 SDT Device resource tree creation in MN-CSE

The procedure to create resource tree for particular SDT Device (deviceLight in this use case) is as follows:

- Device adapter (IPE-AE) sends a CREATE request to gateway (MN-CSE) to create <*flexContainer*> for SDT Device.
- Gateway (MN-CSE) responds with URI of the new <*flexContainer*> for SDT Device.
- Device adapter (IPE-AE) sends a CREATE reqest to gateway (MN-CSE) to create a <*flexContainer*> for a single Module with associated customAttributes (DataPoints). Please note that there is sent one such request per each Module.
- Gateway (MN-CSE) responds with URI for created Module.

- Device adapter (IPE-AE) sends a CREATE request to gateway (MN-CSE) to create <*flexContainer*> for a single Action. The <*flexContainer*> which represents Action is set as a child of <*flexContainer*> which represents Module.
- Gateway (MN-CSE) responds with URI for created Action.

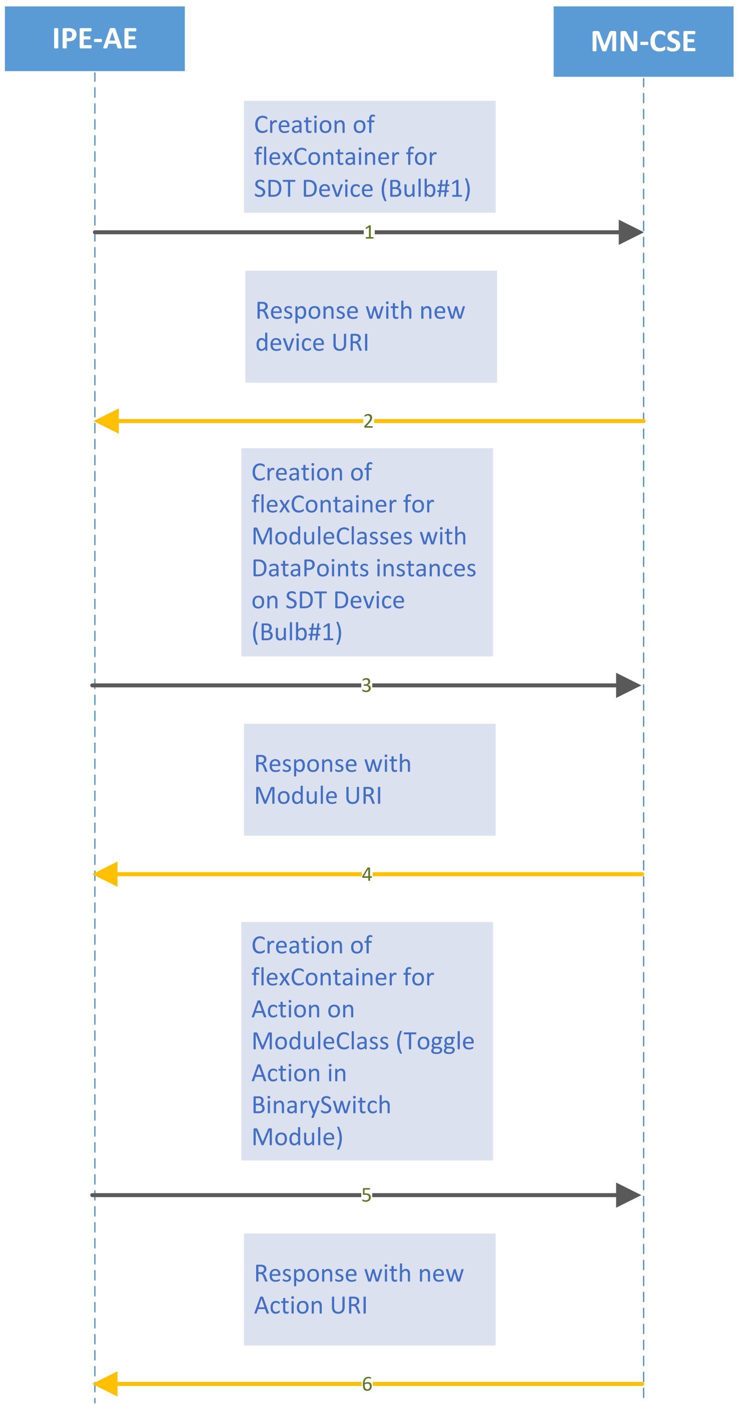

IPE-AE
MN-CSE
Creation of flexContainer for SDT Device (Bulb#1)
1
Response with new device URI
2
Creation of flexContainer for ModuleClasses with DataPoints instances on SDT Device (Bulb#1)
3
Response with Module URI
4
Creation of flexContainer for Action on ModuleClass (Toggle Action in BinarySwitch Module)
5
Response with new Action URI
6

Figure 8.2.2-1: Creation of resource tree for SDT Device

8.3 Call flows from perspective of utility application developer

8.3.1 Application Entity registration in IN/MN-CSE

The procedure to register utility application (ADN-AE-1) with server (IN-CSE) is as follows:

- Utility application (ADN-AE-1) sends registration request to gateway (IN-CSE)
- Server (IN-CSE) responds with status of the registration and Content-Location header for the registered entity

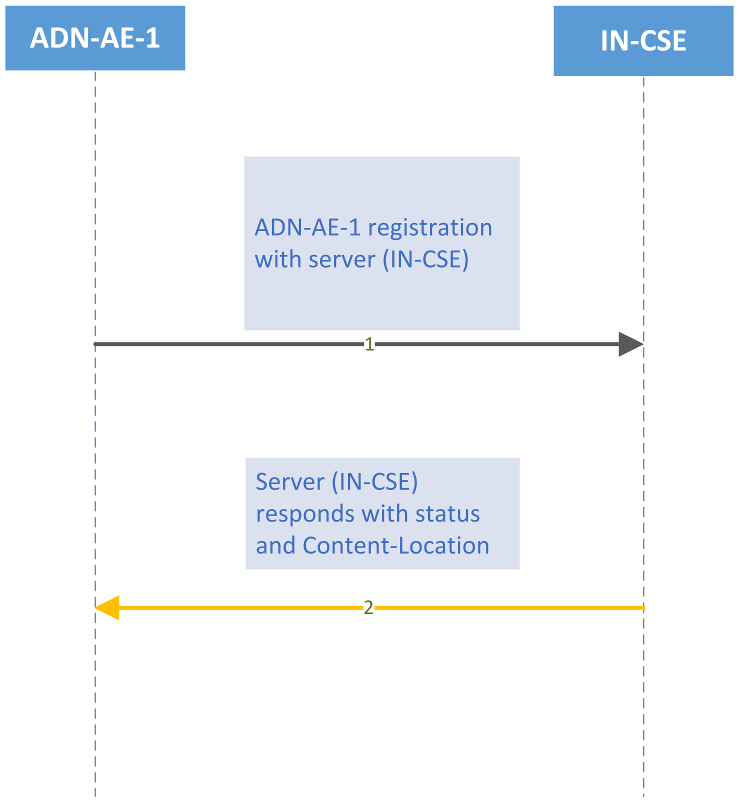

ADN-AE-1
IN-CSE
ADN-AE-1 registration with server (IN-CSE)
1
Server (IN-CSE) responds with status and Content-Location
2

Figure 8.3.1-1: AE registration with CSE request

8.3.2 Discovery Requests

The procedure to discover SDT Device is as follows:

- Utility application (ADN-AE-1) sends a RETRIEVE request to server (IN-CSE) including filter criteria conditions, especially it could be a *containerDefinition* attribute (e.g. org.onem2m.home.devices.deviceLight)
- Server (IN-CSE) responds with list of URIs of the discovered resources, if any (especially URI for deviceLight which represents a bulb).

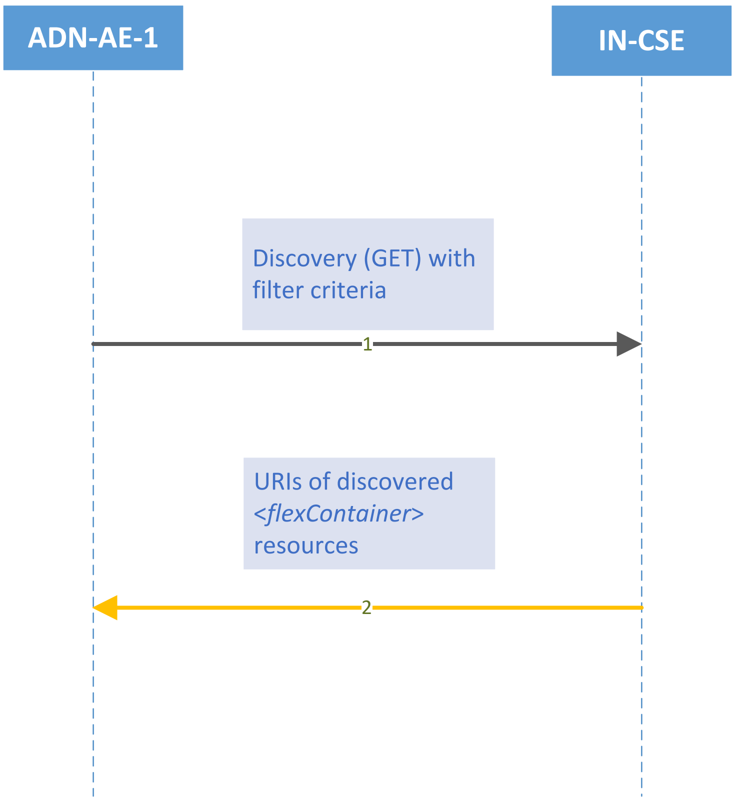

ADN-AE-1
IN-CSE
Discovery (GET) with filter criteria
1
URIs of discovered <flexContainer> resources
2

Figure 8.3.2-1: Discovery of SDT Device with filter criteria

8.3.3 Control & monitor devices

8.3.3.1 Getting URLs for resources

The procedure to retrieve needed URIs for Modules which are associated with particular Device is as follows:

- Utility application (ADN-AE-1) sends a request to server (IN-CSE) with URI of the Device.
- Server (IN-CSE) responds with a list of the Modules URIs.

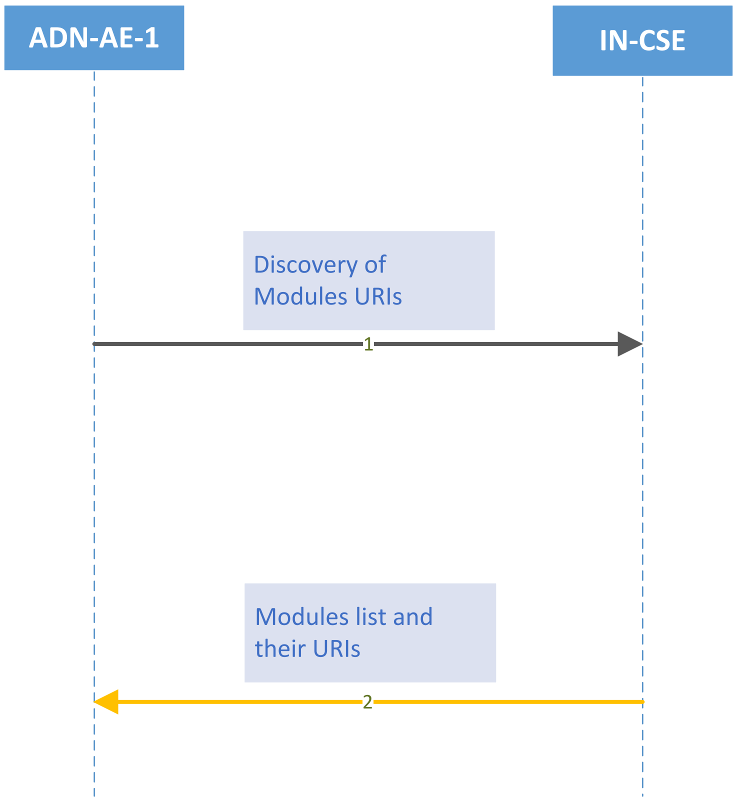


Figure 8.3.3.1-1: Discovery of Modules related to SDT Device

8.3.3.2 DataPoint value retrieving

Retrieving value of a DataPoint is possible only through retrieving all DataPoints which belong to particular ModuleClass.

The procedure to get values of all DataPoints which belong to particular ModuleClass is as follows:

- Utility application (ADN-AE-1) sends a RETRIEVE request to server (IN-CSE) with information which ModuleClass's DataPoints should be retrieved.
- Server (IN-CSE) responds with current values of all DataPoints which belong to the ModuleClass that was requested..

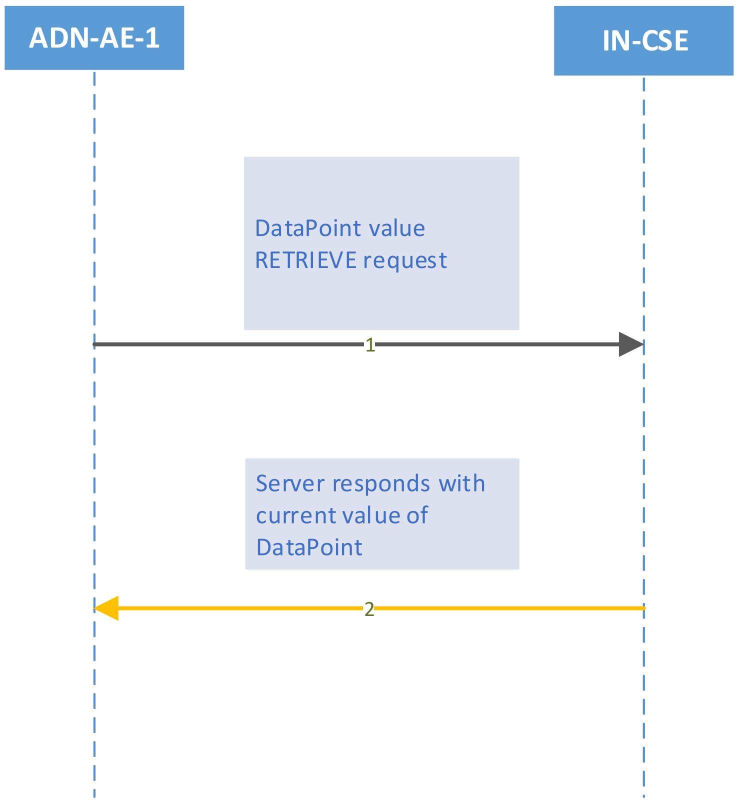

ADN-AE-1
IN-CSE
DataPoint value RETRIEVE request
1
Server responds with current value of DataPoint
2

Figure 8.3.3.2-1: DataPoint retrievement

8.3.3.3 DataPoint value changing

The procedure to change DataPoint value is as follows:

- Utility application (ADN-AE-1) sends an UPDATE request to server (IN-CSE) with <*flexContainer*> which contains the new value of the DataPoint. There is also a possibility to change more than one DataPoint of particular ModuleClass in a single request (in this case the <flexContainer> contains new values of all needed DataPoints.
- Server (IN-CSE) responds with status of DataPoint's UPDATE operation

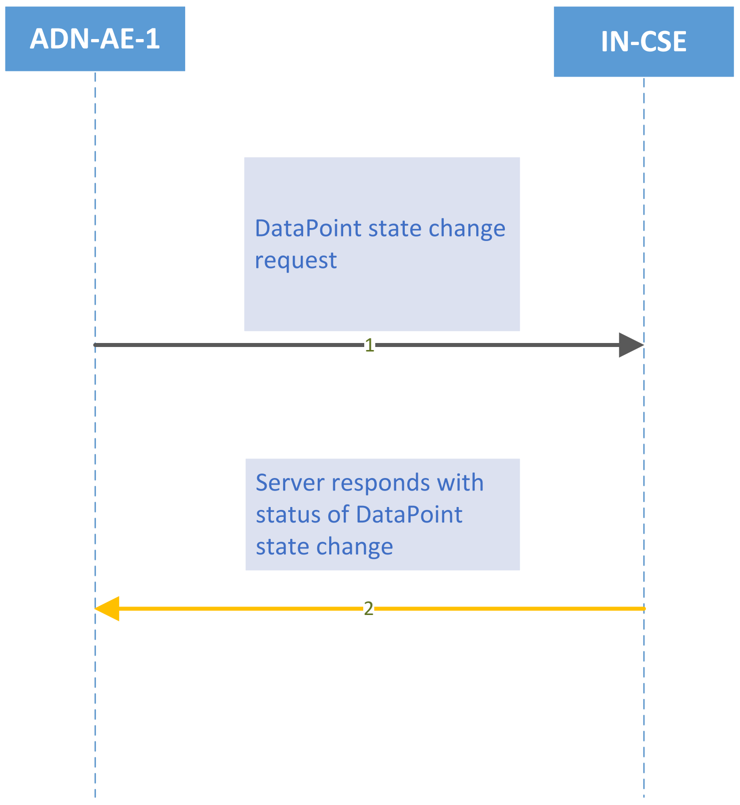

ADN-AE-1
IN-CSE
DataPoint state change request
1
Server responds with status of DataPoint state change
2

Figure 8.3.3.3-1: Change value of a DataPoint

8.3.3.4 Action triggering

The procedure to trigger an Action with no arguments is as follows:

- Utility application (ADN-AE-1) sends an UPDATE request to server (IN-CSE) with empty <*flexContainer*>.
- Server (IN-CSE) responds with status of UPDATE operation

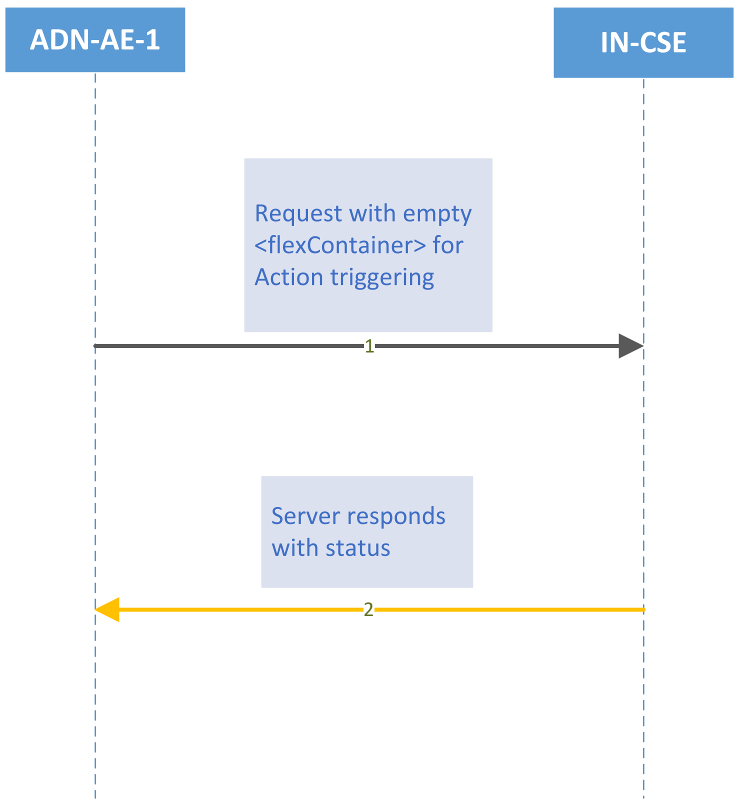

ADN-AE-1
IN-CSE
Request with empty <flexContainer> for Action triggering
1
Server responds with status
2

Figure 8.3.3.4-1: Action triggering procedure

8.3.3.5 Subscription mechanism

Following diagrams show four parts of subscription mechanism: subscription creation, listener's status verification, notification message triggered by DataPoint value change and subsription deletion.

The procedure of creating a <subscription> resource is as follows:

- Utility application (ADN-AE-1) sends a CREATE request to server (IN-CSE).
- Server (IN-CSE) responds with all parameters of created <subscription> resource.

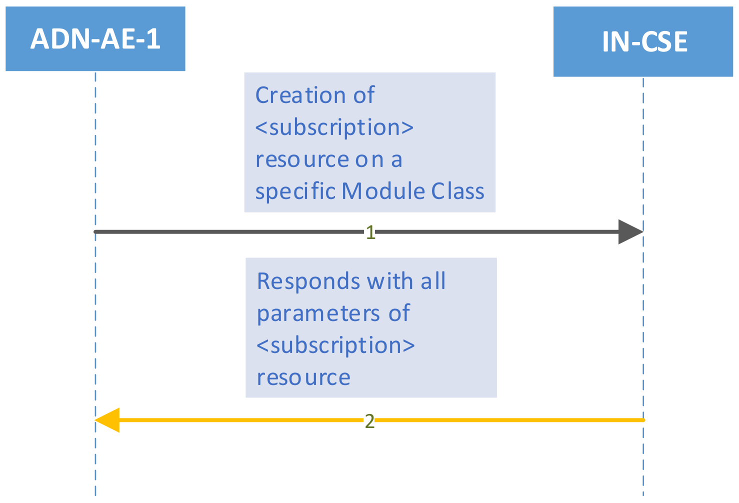

ADN-AE-1
IN-CSE
Creation of <subscription> resource on a specific Module Class
1
Responds with all parameters of <subscription> resource
2

Figure 8.3.3.5-1: Subscription creation procedure

There's a mechanism that allows Server to check listener's availability. The procedure of listener's (ADN-AE-1) status verification is as follows:

- Server (IN-CSE) sends verification request to listener (ADN-AE-1).
- Listener (ADN-AE-1) responds with HTTP 200 OK status.

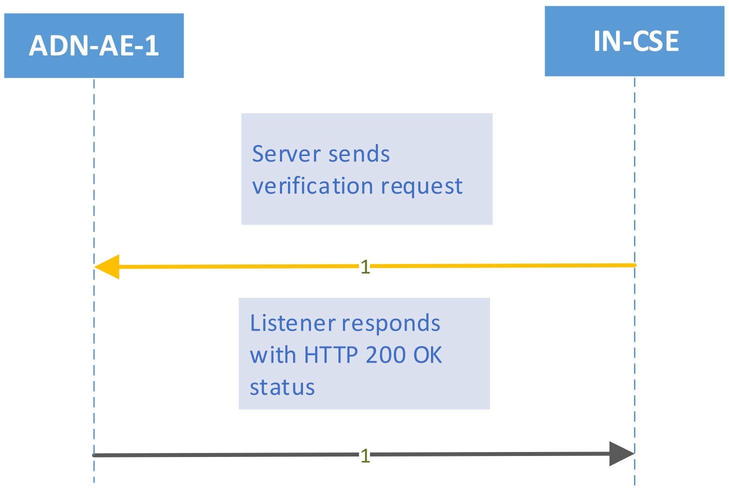
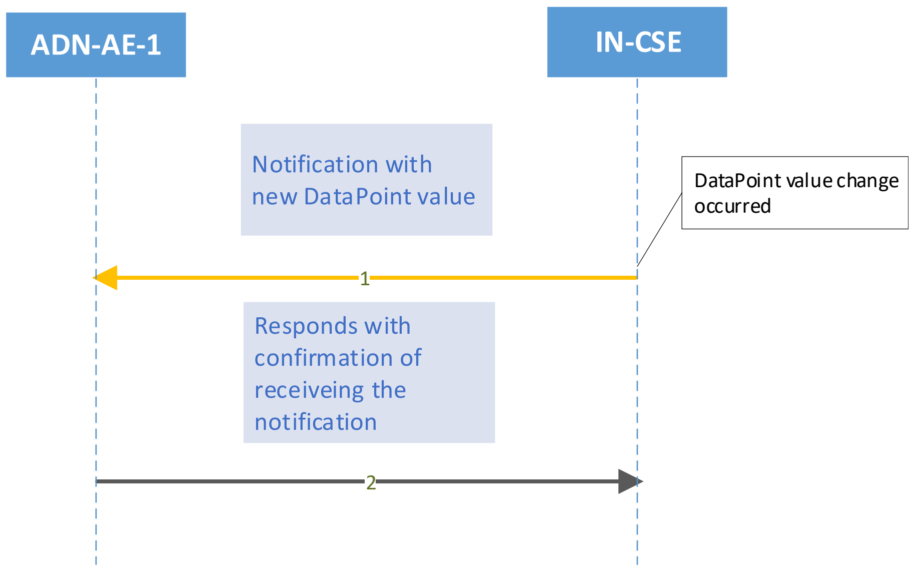

ADN-AE-1
IN-CSE
Server sends verification request
1
Listener responds with HTTP 200 OK status
1

Figure 8.3.3.5-2: Listener's status verification procedure

A change of DataPoint’s value triggers a notification message with new DataPoint value that is sent to listener. The procedure of notification mechanism is as follows:

- DataPoint value change occurs.
- Server (IN-CSE) sends to utility application (ADN-AE-1) notification with new DataPoint value.
- Utility application (ADN-AE-1) sends confirmation of receiving the notification.

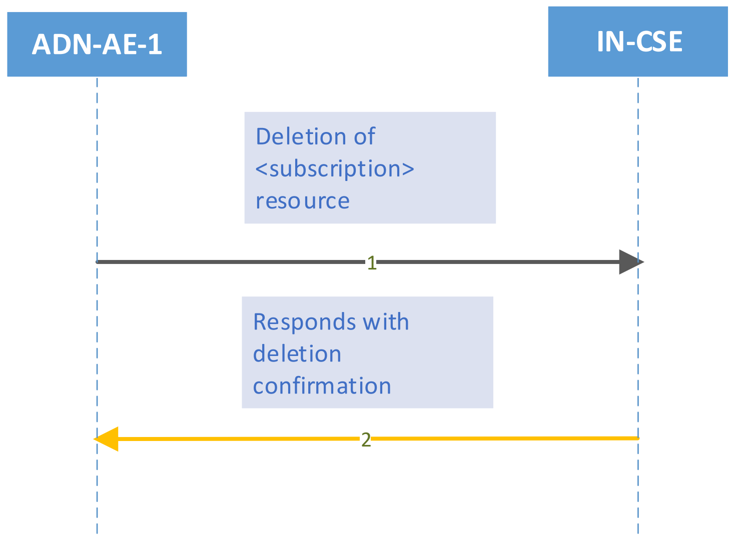


ADN-AE-1
IN-CSE
Notification with new DataPoint value
DataPoint value change occurred
1
Responds with confirmation of receiveing the notification
2

Figure 8.3.3.5-3: Notification mechanism procedure

The procedure of deletion of a <subsription> resource is as follows:

- Utility application (ADN-AE-1) sends a DELETE request to server (IN-CSE).
- Server (IN-CSE) responds with deletion confirmation.



ADN-AE-1
IN-CSE
Deletion of <subscription> resource
1
Responds with deletion confirmation
2

Figure 8.3.3.5-4: Deletion of <subscription> resource procedure

9 Implementation

9.1 Introduction

Clause 9 provides implementation examples from two different perspectives: developer of a device adapter (AE: Inter-working Proxy) in clause 9.3 and developer of a utility application in clause 9.4.

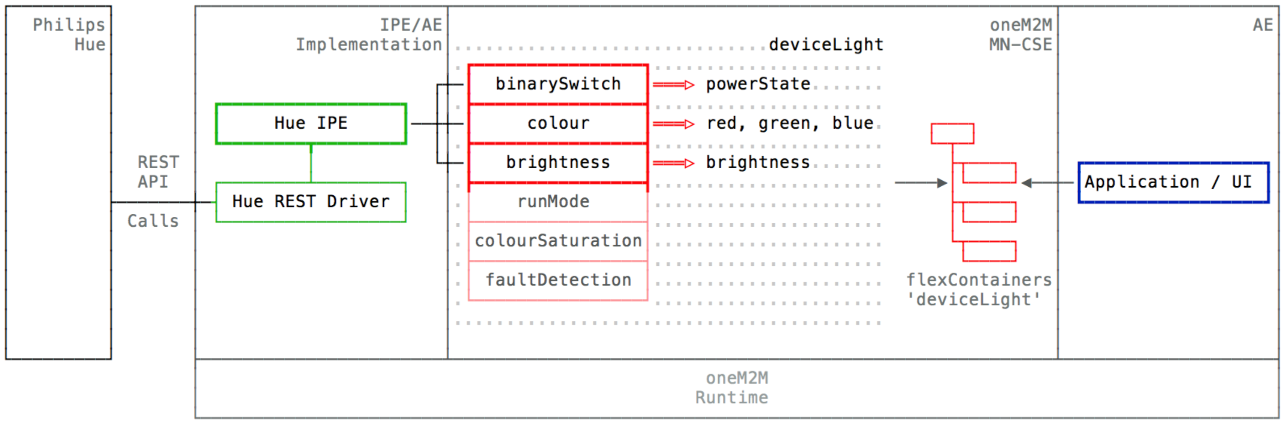


Figure 9.1-1: Implementation of TS-0023 [i.2] Module Classes for the Hue bulb use case

9.2 Assumptions

9.2.0 Introduction

- MN-CSE is registered with IN-CSE
- Philips Hue Bridge and MN-CSE are on the same LAN
- Philips Hue Bulbs are paired with the Philips Hue Bridge
- Philips Hue Bridge is not directly represented in oneM2M resource tree, but through its associated IPE
- Access Control Policy issue isn't discussed here because it's a security aspect which isn't considered in the scope of this developer guide

9.2.1 Addressing for Entities

Each oneM2M entity including AE and CSE are addressable with correct host address that can be IP addresses or FQDN addresses resolved to IP addresses by DNS network services according to addressing rules specified in oneM2M standards.

The IN-CSE and MN-CSE entities presented in this use case are addressable with the following identifiers.

- IN-CSE:
  - CSE-ID: `/in-cse`
  - resourceName of IN-CSE's CSEBase resource: `server`
  - IN-CSE FQDN: `incse.provider.com`
  - IN-CSE HTTP port: `8080`

- MN-CSE：
    - CSE-ID： `/mn-cse`
    - resourceName of MN-CSE's CSEBase resource: **`home_gateway`**
    - MN-CSE FQDN: **`mncse.provider.com`**
    - MN-CSE HTTP port: **`8080`**

The host of Utility Application (ADN-AE-1) has following IP address: 192.168.10.1.

9.3 Developer of the device adapter (IPE-AE)

9.3.0 Introduction

This clause describes implementation process from the IPE-AE developer point of view.

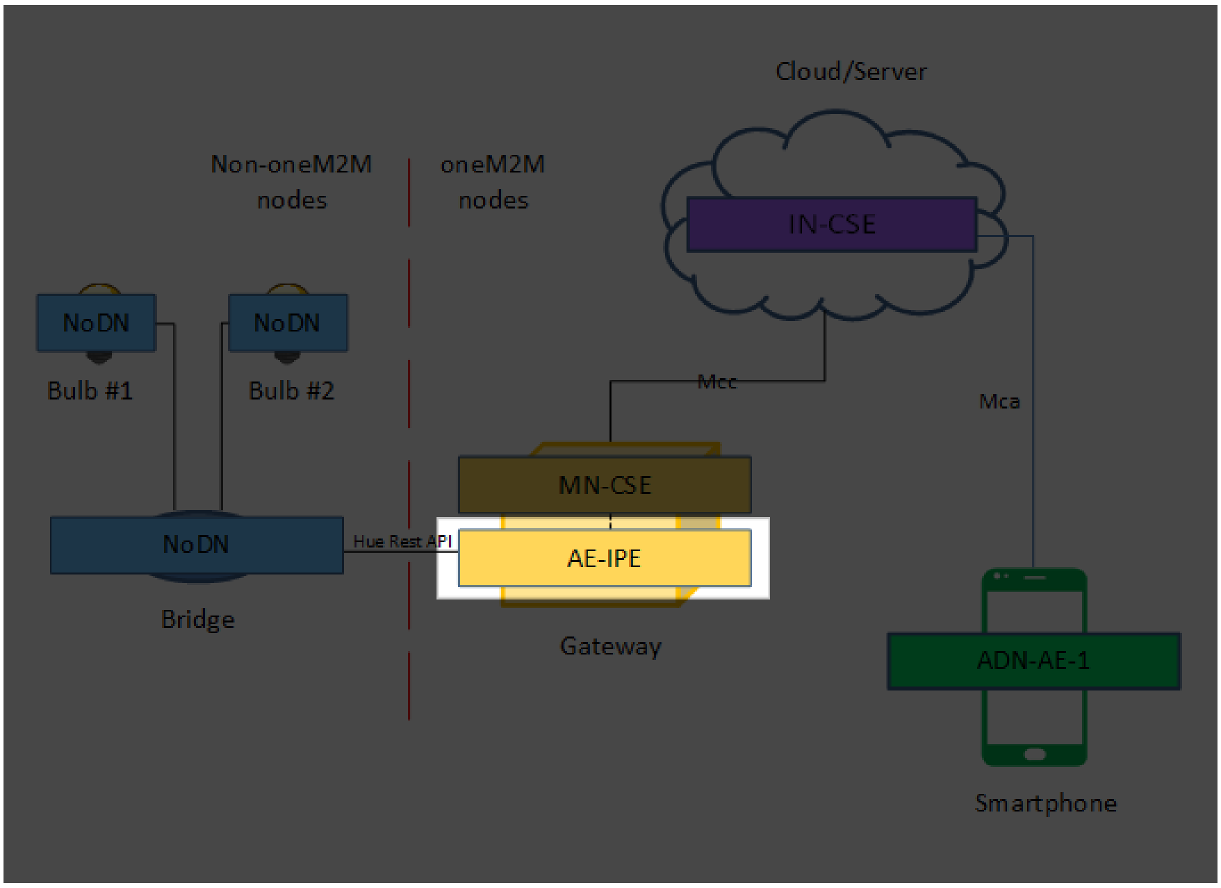


Figure 9.3.0-1 Scope of 9.3 Developer of the device adapter section

9.3.1 IPE-AE registration with MN-CSE

The IPE-AE with MN-CSE registration is shown in the following procedure.

```
POST /home_gateway HTTP/1.1
Host: mncse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml;ty=2
X-M2M-RI: home_gateway-16346
```

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="ipe_ae">
   <api>ipe.lightControl</api>
   <rr>true</rr>
</m2m:ae>

HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: home_gateway-16346
Content-Location: /mn-cse/ae-CAE340304178
```

In response there's Content-Location header which indicates registrated AE address which is necessary to create device resource tree.

AE registration process is comprehensively described in 10.2.2.1 TS-0001 clause.

9.3.2 SDT Device resource tree creation in MN-CSE (Resources registration)

This clause describes creation of device resource tree which consists of Device, ModuleClasses and Actions. Firstly, a Device resource is created. Device is mapped to flexContainer resource, so a flexContainer resource is created.

The following request creates flexContainer resource for deviceLight. In response there is a Content-Location header which indicates the new Device address to beused later in ModuleClass creation.

```
POST /~/mn-cse/ae-CAE340304178 HTTP/1.1
Host: mncse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml;ty=28
X-M2M-RI: home_gateway-12

<?xml version="1.0" encoding="UTF-8"?>
<hd:devLt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="deviceLight">
   <cnd>org.onem2m.home.device.deviceLight</cnd>
</hd:devLt>

HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: home_gateway-12
Content-Location: /mn-cse/DL3404345178
```

Flex Container creation procedure is comprehensively described in 10.2.4.16 TS-0001.

To create ModuleClass on Device, the creation request is sent to Device address taken from Device creation request response. In SDT, ModuleClasses are mapped to flexContainer and DataPoints are mapped to customAttribute. To create binarySwitch ModuleClass, the following request is sent.

```
POST /~/mn-cse/DL3404345178 HTTP/1.1
Host: mncse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml;ty=28
X-M2M-RI: home_gateway-43

<?xml version="1.0" encoding="UTF-8"?>
<hd:binSh xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="binarySwitch">
 <cnd>org.onem2m.home.moduleclass.binaryswitch</cnd>
 <powSe type="xs:boolean">false</powSe>
</hd:binSh>
```

```
HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: home_gateway-43
Content-Location: _/mn-cse/MC43546456
```

In response there's Content-Location header which indicates created ModuleClass address which is used later to create Toggle Action and to modificate state of particular DataPoint.

In SDT, Actions are also mapped to FlexContainer. To add new Action to ModuleClass, the the following request is sent to the ModuleClass address.

```
POST /~/mn-cse/MC43546456 HTTP/1.1
Host: mncse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml;ty=28
X-M2M-RI: home_gateway-44

<?xml version="1.0" encoding="UTF-8"?>
<hd:toggle xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="toggle">
  <cnd>org.onem2m.home.moduleclass.binaryswitch.toggle</cnd>
</hd:toggle>

HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: home_gateway-44
Content-Location: /mn-cse/AC-54783722
```

In response there's Content-Location header which indicates new Action address which is used later to trigger Action (described in clause 9.4.3).

9.3.3 Implementation of abstract SDT Device example: deviceLight

This clause presents a “pseudo code” implementation of abstract SDT Device based on deviceLight example. Please note that this pseudo-code is similar to Java but should be considering just an example, it isn’t attend to be compilable.

DeviceLight class is an implementation of deviceLight model, it inherits from Device class and has 8 fields relevant for 8 ModuleClasses of deviceLight model. DeviceLight has a contructor which creates objects for ModuleClasses and defines Name and Domain.

```
/**

 *

 * Stores information about name and domain of the device

 *

*/

class Device{
     string name;
     string domain;

     public Device(string name, string domain){
        name = name;
        domain = domain;
     }
```

```
}

/**
  * DeviceLight class is an implementation of deviceLight model, it inherits from Device
  * class, and has 8 fields relevant for 8 ModuleClasses of deviceLight model.
  * DeviceLight has a contructor which creates objects for ModuleClasses and defines
  * Name and Domain.
  */
class DeviceLight extends Device{
     FaultDetection faultDetection;
     BinarySwitch binarySwitch;
     RunMode runMode;
     Colour colour;
     ColourSaturation colourSaturation;
     Brightness brightness;

     public DeviceLight(string name, string domain){
        super(name, domain);
     }
}


/**
* Module consists of 3 fields which are relevant to DataPoints parameters in SDT.
*/
class Module{
   boolean optional;
   string name;
   string domain;

   public Module(string name, string domain, boolean optional){
      this.name = name;
      this.optional = optional;
      this.domain = domain;
   }
}


/**

* BinarySwitch class is considered as an example of ModuleClass implementation,

* it inherits from Module. It has fields relevant for DataPoints and Actions

* which are defined in SDT data model. Constructor of BinarySwitch creates powerState

* DataPoint. BinarySwitch class has also setToggle function which allows to set toogle

* action which is not mandatory.

*/
class BinarySwitch extends Module {
     BooleanDataPoint powerState;
     Toggle toggle;

     public BinarySwitch(string moduleName, string moduleDomain, BooleanDataPoint
powerStateArgument){
        super(moduleName, moduleDomain, false);

        powerState = powerStateArgument;
        powerState.optional = false;
        powerState.readable = true;
        powerState.writeable = true;
        powerState.documentation = "The current status of the binarySwitch. "True"
      indicates turned-on, and "False" indicates turned-off";
      }

      void setToggle(Toggle toggleArgument){
        toggle = toggleArgument;
        toggle.optional = true;
      }

}
```

/**

 * Colour extends Module and has 3 fields relevant to Colour ModuleClass DataPoints.

* In the constructor of Colour class there are DataPoints assingments with their

* parameters. There's also abstract setColour method which allows to set

* all colours (R,G,B) at once.

```
*/
class Colour extends Module {
   IntegerDataPoint red;
   IntegerDataPoint green;
   IntegerDataPoint blue;

   public Colour(string moduleName, string moduleDomain, IntegerDataPoint redArgument,
IntegerDataPoint greenArgument, IntegerDataPoint blueArgument){

      super(moduleName, moduleDomain, true);

      red = redArgument;
      green = greenArgument;
      blude = blueArgument;

      red.optional = false;
      red.readable = true;
      red.writeable = true;
      red.documentation = "";


      green.optional = false;
      green.readable = true;
      green.writeable = true;
      green.documentation = "";


      blue.optional = false;
      blue.readable = true;
      blue.writeable = true;
      blue.documentation = "";
   }

/**
```

* This method allows to set all colours (R,G,B) in a single call.

* Since it is an abstract method it has to be implemented by developer according to

* specific technology which the device uses.

*/

```
   public abstract setColour(int R, int G, int B);

}


/**

* Toggle class consists of 3 fields which are relevant to SDT Action parameters and

* toggle() method which determines Toggle Action.

*/

class Toggle {
   boolean optional = true;
   string documentation;
   string name;

   public abstract void toggle(){}

   public Toggle(boolean optional, string documentation, string name){
      this.optional = optional;
      this.documentation = documentation;
      this.name = name;
   }
}

/**

* DataPoint class constis of 6 fields relevant to DataPoint SDT parameters.

* In the constructor there's name and type assingments. The rest of paremeters are

* assingments in classes which inherits from DataPoint class.

*/

class DataPoint{
      string name;
      string type;
      boolean optional;
      bool readable;
      bool writeable;
      string documentation;

      public DataPoint(string nameArgument, string typeArgument){
         name = nameArgument;
         type = typeArgument;
      }
}

/**

* IntegerDataPoint class consists of two abstract methods getValue() and setValue()

* which are implemented in HueLight class in 9.3.4 clause.

* Notice there is DataPoint constructor called in IntegerDataPoint constructor.

*/
public abstract class IntegerDataPoint extends DataPoint{

      public IntegerDataPoint(string name){
         super(name, "Integer");
```

```
    }

    abstract int getValue(){};

    abstract void setValue(int value){};
}

/**
* BooleanDataPoint class consists of two abstract methods getValue() and setValue()
* which are implemented in HueLight class in 9.3.4 clause.
* Notice there is DataPoint constructor called in BooleanDataPoint constructor.
*/
public abstract class BooleanDataPoint extends DataPoint{

    public BooleanDataPoint(string name){
      super(name, "Boolean");
    }

    abstract boolean getValue();

    abstract void setValue(boolean value);

}
```

9.3.4 Instance of SDT Device example: Hue Light

This section presents a “pseudo code” implementation of IPE for particular device: Hue Light. Please note that this pseudo-code is similar to Java but should be considering just an example, it isn’t attend to be compilable.

```
/**
* HueLight class inherits from DeviceLight class. In the contructor of this class
* DataPoints are created with all needed abstract methods implementation
* specific for technology (Hue Light in this use case) the device uses. It means that
* each DataPoint (e.g. powerState in BinarySwitch) has implementation of each abstract
* method which consists of Hue API calls to control Hue Light.
*/
class HueLight extends DeviceLight{

  public HueLight(string name, string domain){
    super(name, domain);
  }

  BinarySwitch = new BinarySwitch("binarySwitch", domain,
new BooleanDataPoint("powerState"){
      boolean getValue(){
        HttpRequest httpRequest;
        httpRequest.setURL("http://<bridge_ip_address>/api/1028d66426293e821ecfd9ef1a
        0731df/lights/1/state");
        httpRequest.setMethod("GET");
        bool state = parseOnOff(httpRequest.sendHTTPRequest());
        return state;
      }

      void setValue(boolean value){
        HTTPRequest httpRequest;
        httpRequest.setURL("http://<bridge_ip_address>/api/1028d66426293e821ecfd9ef1a
        0731df/lights/1/state");
        httpRequest.setBody("{\"on\":" + value + "}");
        httpRequest.setMethod(“PUT”);
```

```
            httpRequest.sendHTTPRequest();
         }
      });

   Colour = new Colour("colour", domain, new IntegerDataPoint("red"){

         int getValue(){
            HttpRequest httpRequest;
            httpRequest.setURL(http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setMethod("GET");
            int red = parseRed(parseColor(httpRequest.sendHTTPRequest()));
            return red;
         }
         void setValue(int value){

            HSVColour hsv = new HSVColour();
            hsv.convertFromRGB(value, 0, 0);

            HTTPRequest httpRequest;
            httpRequest.setURL("http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" +
            hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}"});
            httpRequest.setMethod("PUT")
            httpRequest.sendHTTPRequest();
         }
   }, new IntegerDataPoint("green"){
         int getValue(){
            HttpRequest httpRequest;
            httpRequest.setURL(http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setMethod("GET");
            int green = parseGreen(parseColor(httpRequest.sendHTTPRequest()));
            return green;
         }

         void setValue(int value){
            HSVColour hsv = new HSVColour();
            hsv.convertFromRGB(0, value, 0);

            HTTPRequest httpRequest;
            httpRequest.setURL("http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" +
            hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}"});
            httpRequest.setMethod("PUT")
            httpRequest.sendHTTPRequest();
         }
   }, new IntegerDataPoint("blue"){
        int getValue(){
            HttpRequest httpRequest;
            httpRequest.setURL(http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setMethod("GET");
            int blue = parseBlue(parseColor(httpRequest.sendHTTPRequest()));
            return blue;
        }
        void setValue(int value){
            HSVColour hsv = new HSVColour();
            hsv.convertFromRGB(0, 0, value);
            HTTPRequest httpRequest;
            httpRequest.setURL("http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" +
            hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}"});
            httpRequest.setMethod("PUT")
            httpRequest.sendHTTPRequest();
        }

    }){
        void setColour(int R, int G, int B){
            HSVColour hsv = new HSVColour();
            hsv.convertFromRGB(R, G, B);

            HTTPRequest httpRequest;
            httpRequest.setURL("http://<bridge ip
            address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");
            httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" +
            hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}"});
            httpRequest.setMethod("PUT")
            httpRequest.sendHTTPRequest();
        }
    };
}
```

/**

* SDT Colour model uses RGB colour values and Hue Light device uses HSV model.

* HSVColour class allows to convert RGB to HSV with its convertFromRGB methods.

* This class and method isn't a part of SDT data model.

*/

```
class HSVColour{

int hue;
int sat;
int value;

   void convertFromRGB(int R, int G, int B){
      //Algorithm that converts RGB values to HSV values should be implemented here
   }
}
```

9.4 Developer of the utility application

9.4.0 Introduction

Clause 9.4 describes registration of AE for utility application, discovery process and controlling and monitoring devices.

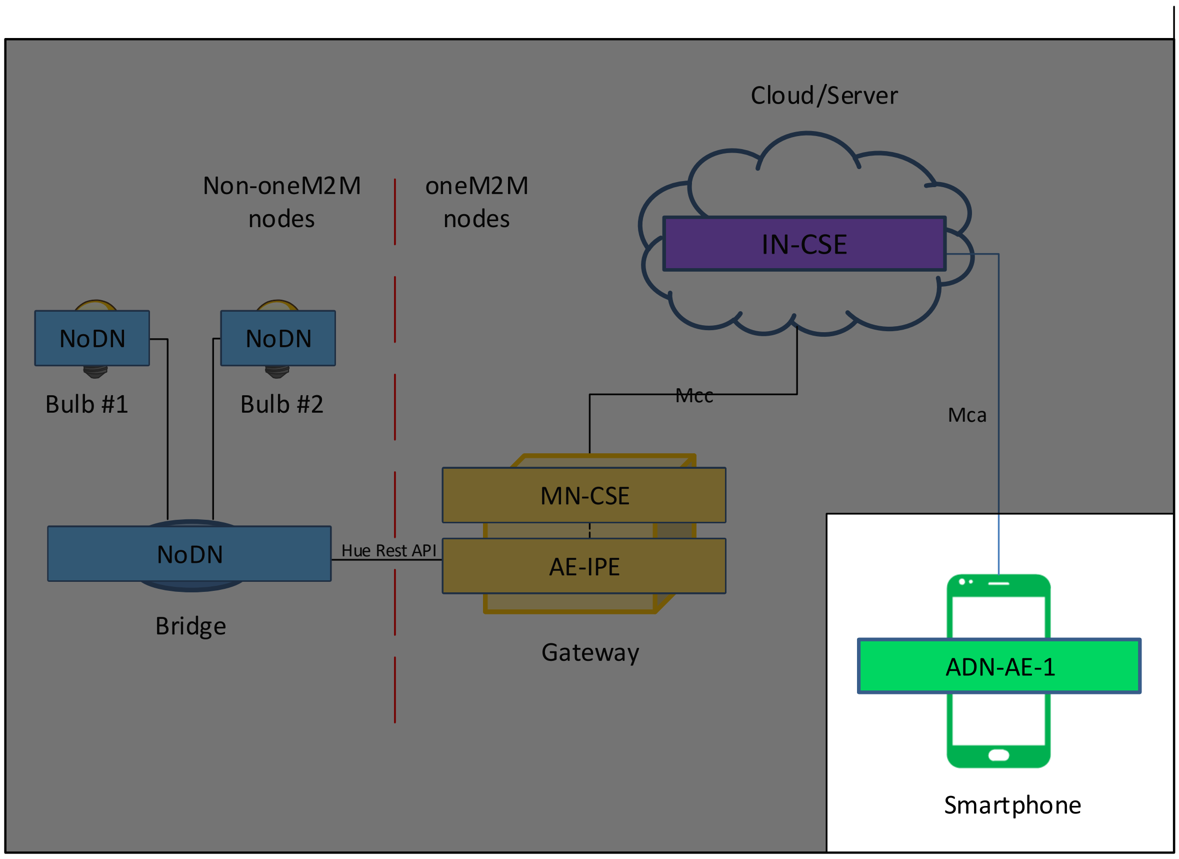


Figure 9.4.0-1 Scope of 9.4 Developer of the utility application section

9.4.1 Application Entity registration in IN/MN-CSE

The ADN-AE with IN-CSE registration is shown in the following procedure.

```
POST /server?rcn=0 HTTP/1.1
Host: incse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml;ty=2
X-M2M-RI: server-1234

<?xml version="1.0" encoding="UTF-8"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="smartphone_ae">
  <api>incse.lightControlApp</api>
  <rr>true</rr>
</m2m:ae>

HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: server-1234
Content-Location: /in-cse/ae-CAE345
```

In response Content-Location header there's an address of registered Application Entity.

AE registration process is comprehensively described in 10.2.2.1 TS-0001 clause.

9.4.2 Discovery proccess

Clause 9.4.2 describes discovery proccess which uses filter criteria to limit the scope of returned information.

Folowing response searches for resources with cnd(Container Definition) set to org.onem2m.home.device.deviceLight so it gives in response list of urils of light devices. Uril is neccessary to control particular device. Fu (filter usage) parameter value set to '1' indicates this is normal resource discovery request so only the addresses of the child resources are returned.

```
GET  /server?fu=1&cnd=org.onem2m.home.device.deviceLight  HTTP/1.1
Host: incse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml
X-M2M-RI: server-1234


HTTP Response:

200 OK
X-M2M-RSC: 2000
X-M2M-RI: server-1234
Content-Type: application/xml

<m2m:uril xmlns:m2m="http://www.onem2m.org/xml/protocols">
mn-cse/DL3404345178
</m2m:uril>
```

Discovery process is comrehensively described in TS-0001 6.2.5 and 10.2.6 clause.

9.4.3 Control & monitor devices

9.4.3.0 Introduction

Clause 9.4.3 describes how to control and monitor SDT devices registered in oneM2M network.

9.4.3.1 Getting URLs for resources

First there is a need to get a url of ModuleClass resource. ModuleClass is a child of deviceLight, so there is a need to send a request to device address which returns a list of it's children in response. In this particular case there're two: binarySwitch and toggle. Each of them should be controlled by requests sent on given addresses.

Following request gives a list of deviceLight children (ModuleClassess or Actions) in response with their urils.

```
GET /~/mn-cse/DL3404345178?rcn=6
HTTP/1.1
Host: incse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml
X-M2M-RI: server-12345

HTTP Response:

200 OK
X-M2M-RSC: 2000
X-M2M-RI: server-12345
Content-Location: /in-cse/ae-CAE345

<?xml version="1.0" encoding="UTF-8"?>
<m2m:fcnt xmlns:m2m="http://www.onem2m.org/xml/protocols">
    <ch rn="binarySwitch" ty="28">/mn-cse/MC43546456 </ch>
    <ch rn="toggle" ty="28">/mn-cse/AC-54783722 </ch>
</m2m:fcnt>
```

Here is data from previous request answer which is needed:

```
<ch rn="binarySwitch" ty="28">/mn-cse/MC43546456 </ch>
<ch rn="toggle" ty="28">/mn-cse/AC-54783722 </ch>
```

9.4.3.2 DataPoint value retrieving

To monitor powerState , a RETRIEVE request to binarySwitch resource (`/mn-cse/MC43546456`) is sent as following. In response there is xml payload which contains whole binarySwitch representation.

Remark: Retrieving value of a DataPoint is possible only through retrieving all DataPoints which belong to particular ModuleClass. The set of all ModuleClass's DataPoints is sent in response. .

```
GET /~/mn-cse/MC43546456?rcn=1 HTTP/1.1
Host: incse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml
X-M2M-RI: server-12346

HTTP Response:

200 OK
X-M2M-RSC: 2000
X-M2M-RI: server-12346

<?xml version="1.0" encoding="UTF-8"?>
<hd:binSh xmlns:m2m="http://www.onem2m.org/xml/protocols">
  <powSe type="xs:boolean">true</powSe>
</hd:binSh>
```

9.4.3.3 DataPoint value changing

To change powerState value, a request to binarySwitch resource (`/mn-cse/MC43546456 `) is sent with xml (or json) representation of new powerState value. PUT request is used because whole resource is updated.

Following requests changes power State Data Point to 'true'.

```
PUT /~/mn-cse/MC43546456 HTTP/1.1
Host: incse.provider.com:8080
X-M2M-Origin: Soriginator
Content-Type: application/xml
X-M2M-RI: server-12346

<?xml version="1.0" encoding="UTF-8"?>
<hd:binSh xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="smartphone_ae">
  <powSe type="xs:boolean">true</powSe>
</hd:binSh>

HTTP Response:

204 Updated
X-M2M-RSC: 2004
X-M2M-RI: server-12346
```

9.4.3.4 Action triggering

To trigger Toggle action a null content parameter is sent to toggle resource address. (see TS-0023, 6.2.4 Clause, Rule 3-4).

Following request sends null content and triggers Toggle action.

```
PUT /~/mn-cse/AC-54783722 HTTP/1.1
Host: incse.provider.com:8080
```

```
        X-M2M-Origin: Soriginator
        Content-Type: application/xml
        X-M2M-RI: server-12346

        <?xml version="1.0" encoding="UTF-8"?>
        <hd:toggle xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="smartphone_ae">
        </hd:toggle>

        HTTP Response:

        204 Updated
        X-M2M-RSC: 2004
X-M2M-RI: server-12346
```

9.4.3.5 Subscription mechanism

Following request creates subscription on binarySwitch resource (`mn-cse/MC43546456)` which allows to send notifications after its state change. There are two mandatory parameters.

NotificationURI parameter is represented by <nu> xml tag and indicates the address of the device which is supposed to be notificated about ouccured changes. In our case smartphone IP address is used.

Remark: In our use case smartphone IP address is used as a receiver of notifications but in other usage there can be used notification server with static IP address assigned. In such use case this server would be responsible to send notification directly on smartphone address.

NotificationContentType is represented by <nct> xml tag and indicates the type of notification. Its value isone of the following:
- 1: all atributes
- 2: modified attributes,
- 3: resourceID

In our example <nct> is set to 2.

In response there is a whole set of subscription resource parameters.

```
        POST /~/mn-cse/MC43546456 HTTP/1.1
        Host: incse.provider.com:8080
        X-M2M-Origin: Soriginator
        Content-Type: application/xml
        X-M2M-RI: server-12346


        <m2m:sub xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="sub_45346345">
            <nu>192.168.10.1</nu>
            <nct>2</nct>
        </m2m:sub>

        HTTP Response

        HTTP/1.1 200 OK
        X-M2M-RSC: 2000
        X-M2M-RI: server-12345

        <rn>sub_45346345</rn>
        <ty>23</ty>
        <ri>/~/mn-cse/MC43546456/sub-45346345</ri>
        <pi>/~/mn-cse/MC43546456/fcnt-45352623</pi>
        <ct>20170802T143815</ct>
        <lt>20170802T143815</lt>
        <nu>192.168.10.1</nu>
        <nct>2</nct>
```

After subscription creation, server starts to send verification requests to listener periodically. Verification request has <vrq> attribute set to true. Listener answers with HTTP 200 status to confirm that it is still available. Verification request and listener's response is as follows:

```
POST 192.168.10.1 / HTTP/1.1
Host: 192.168.10.1
X-M2M-Origin: Soriginator
X-M2M-RI: notif-12345
Content-Type: application/vnd.onem2m-ntfy+xml

<?xml version="1.0" encoding="UTF-8"?>
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
    <vrq>true</vrq>
    <sud>false</sud>
</m2m:sgn>

HTTP Response

HTTP/1.1 200 OK
X-M2M-RSC: 2000
X-M2M-RI: notif-12345
```

Here is the example of notification which is supposed to be sent after powerState value change. Notification is sent to address which was set in subsription creation request and contains representation of changed DataPoints of binarySwitch resource.

```
POST 192.168.10.1 / HTTP/1.1
    Host: 192.168.10.1
    X-M2M-Origin: Soriginator
    X-M2M-RI: notif-12345
    Content-Type: application/vnd.onem2m-ntfy+xml

    <?xml version="1.0" encoding="UTF-8"?>
    <m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
      <nev>
        <rep>
         <powSe type="xs:boolean">true</powSe>
        </rep>
        <rss>1</rss>
      </nev>
      <sur> mn-cse/MC43546456/sub-45346345</sur>
    </m2m:sgn>

HTTP Response

HTTP/1.1 200 OK
X-M2M-RSC: 2000
X-M2M-RI: notif-12345
```

Following request deletes subscription on binarySwitch.

```
DELETE /~/mn-cse/MC43546456/sub-45346345 HTTP/1.1
Host: incse.provider.com:8080
X-M2M-Origin: Soriginator
Accept: application/xml

HTTP Response

HTTP/1.1 204 No Content
```

10 Conclusion

A remote light control and monitor use case is proposed to describe IPE and SDT aspects of oneM2M standard. The developer guide described these ascpects from two different perspectives: developer of a device adapter (AE: Interworking Proxy) and developer of an utility application. The current use case is done by following the high level procedures such as registration of AE with CSE, <flexContainer> resources creation, <flexContainer> retrieveal, <flexContainer> update, subscription and notifications.

History

Publication history		
V2.0.0	12-Mar-2018	Release 2A - Publication