Middleware for the IoT: TP1

Interact with the oneM2M RESTful architecture using Eclipse OM2M

LIEVRE Agathe

NGUYEN Assia

5ISS

2021/2022

Objective

In this class, we learned how to use the Eclipse OM2M platform. We first configured and launched the platform. We then used Postman as a REST client to create http requests to both the OM2M server (IN-CSE) and the OM2M Gateway (MN-CSE). It allowed us to create a full ressource tree and control the associated access control policies.

Access Control Management in oneM2M: exercise

In this exercise, the goal was to create an ACP in the already created ressource tree and then attach it to a data container. Then, we gave access to a monitoring application and a sensor application that have different access control operations. The monitoring application can retrieve the data whereas the sensor application can retrieve and create data.

To create the ACP we sent the following POST request to the gateway in json. In the header we specified the originator and the type of the entity. Here the type is 1 because it is an ACP.

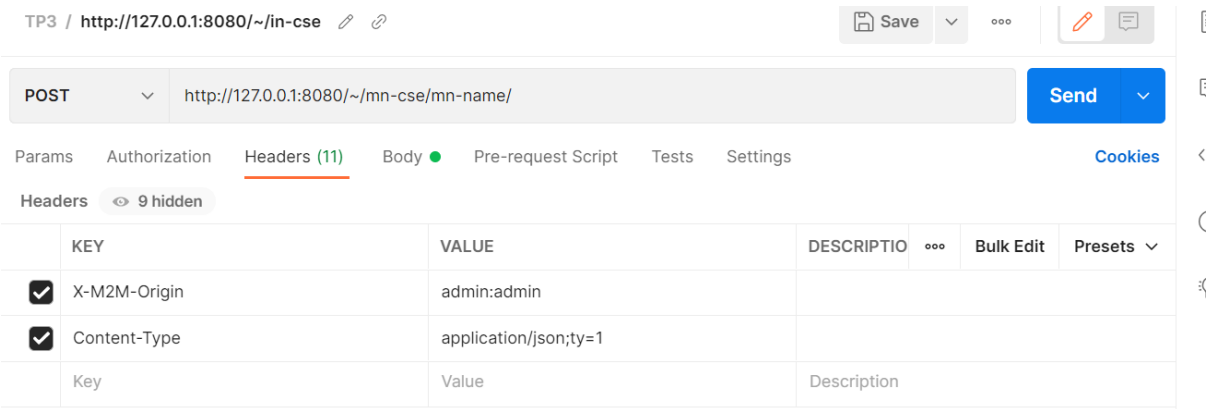

Figure 1: http POST request header for ACP creation

The body of the request is below. In this script we defined the access control policy. We defined the privilege, in other words, who has access to the linked AE resources. We chose to give all the access rights to the admin by puting 63 in the acop field. Indeed, according to the following table, the sum of all the right's value equals 63.

Access Control Operation	Value
CREATE	1
RETRIEVE	2
UPDATE	4
DELETE	8
NOTIFY	16
DISCOVERY	32

Figure 2: Table of the access control operations and associated values

In the self-privilege field, to specify who has access to the ACP itself, we also gave the admin all the access rights.

```
{
    "m2m:acp": {
        "rn": "my_acp",
        "pv": {
            "acr": [
                {
                    "acor": [
                        "admin:admin"
                    ],
                    "acop": 63
                }
            ]
        },
        "pvs": {
            "acr": [
                {
                    "acor": [
                        "admin:admin"
                    ],
                    "acop": 63
                }
            ]
        }
    }
}
```

Figure 3: Script in json for ACP creation

The result on the OM2M platform can be seen below. The ACP was created and possesses the pv and pvs we defined.

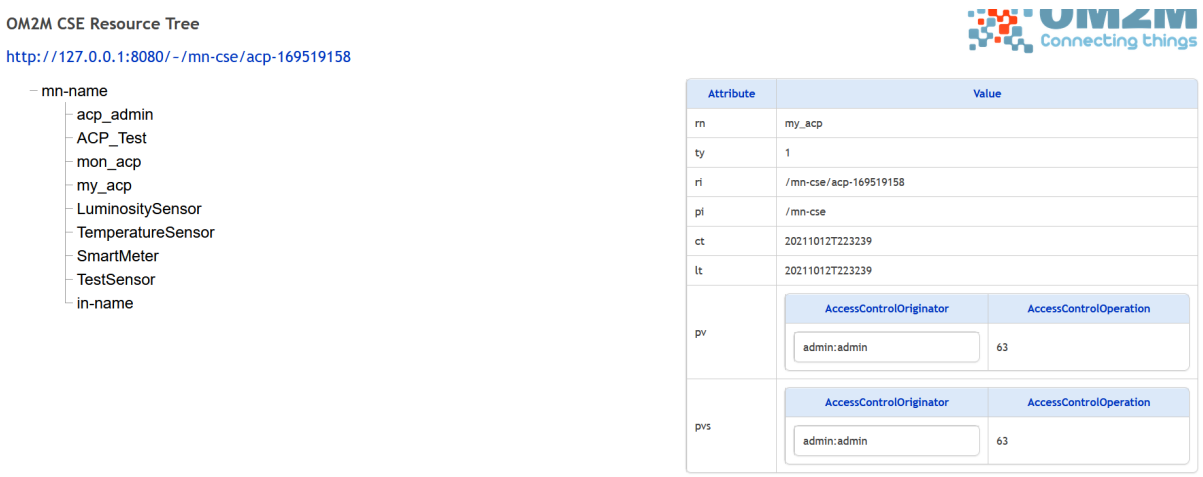

Figure 4: OM2M ressource tree with the ACP attributes

Scenario

The objective of this scenario was to simulate several sensors. Then, we had to register an application that will be used as a monitor using the Subscription mechanism.

First, we created 3 AE on the MN with the following names: SmartMeter, LuminositySensor and TemperatureSensor.

For example, to create the SmartMeter, we used this request:

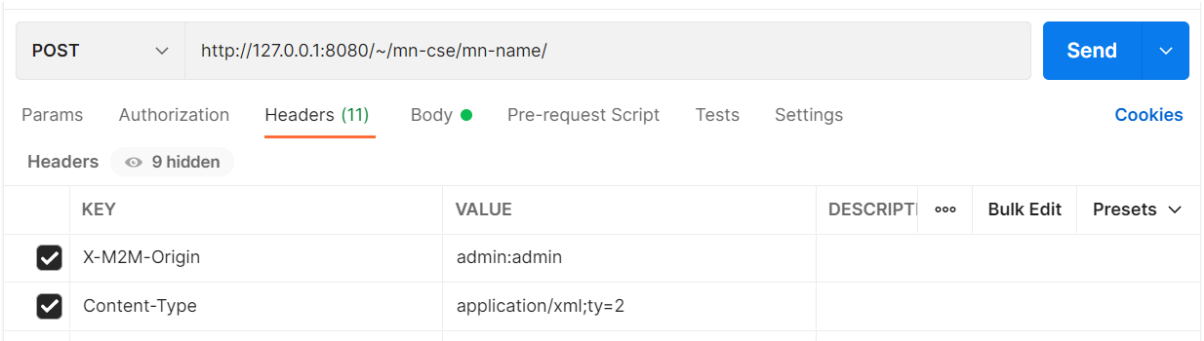

Figure 5: http POST request header for AE creation

The type defined in the header is 2 because we want to create an AE. We want this AE to be associated with the ACP we defined earlier. To do so, we added a line <acpi> with the“my_acp” ID.

```
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="SmartMeter" >
    <acpi>/mn-cse/acp-169519158</acpi>
    <api>app-smart</api>
    <lbl>Type/meter Category/smart Location/home</lbl>
    <rr>false</rr>
</m2m:ae>
```

Figure 6: Script in XML for AE creation

The other AE’s requests have the same format.

Then, within each AE, we created two containers: DESCRIPTOR and DATA. Because they are containers, they have a type 3.The following figures show the header and the body of the request for the DESCRIPTOR and the DATA container of the SmartMeter.

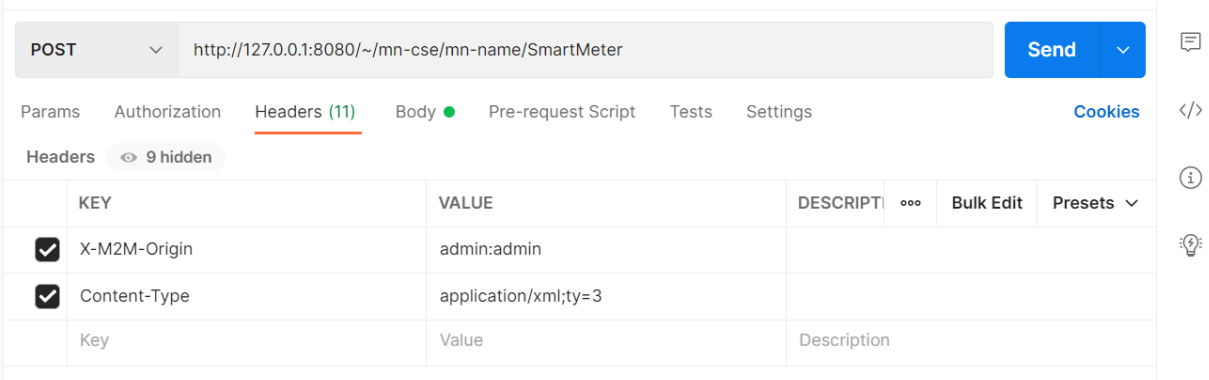

Figure 7: http POST request header for container creation (Descriptor)

```
1  <m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="DESCRIPTOR">
2  </m2m:cnt>
```

Figure 8: Script in XML for container creation (Descriptor)

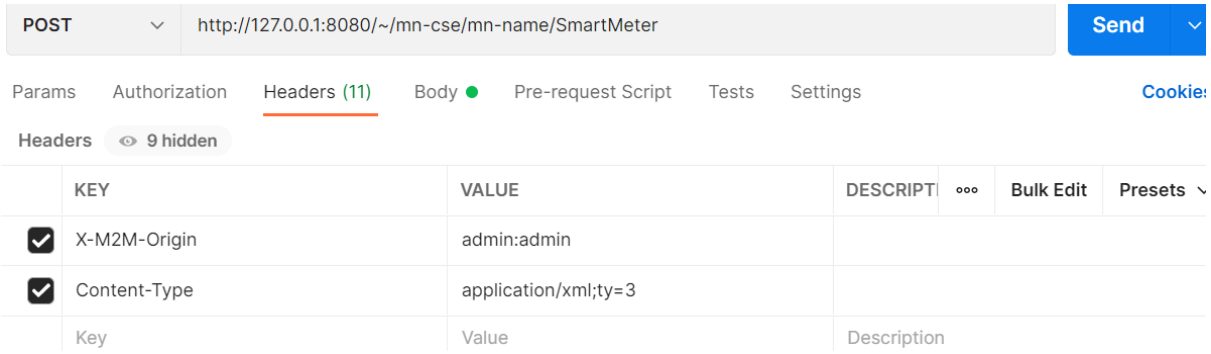

Figure 9: http POST request header for container creation (Data)

```
<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="DATA">
</m2m:cnt>
```

Figure 10: Script in XML for container creation (Data)

After creating the containers, we made one content instance (type 4) for the DESCRIPTOR container and one for the DATA container. We translated the oBIX model given to us in the instructions into a XML request. For example, the requests for the content instance of the SmartMeter's DESCRIPTOR and DATA container are as follows:

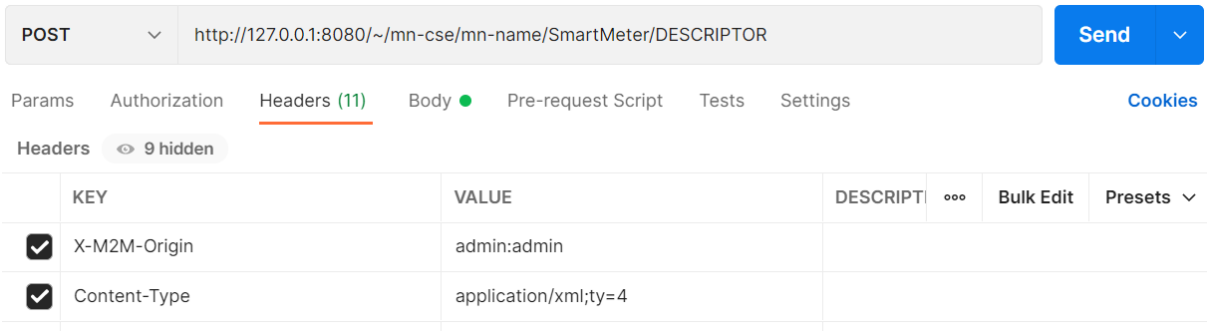

Figure 11: http POST request header for instance creation

```
<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/protocols">
    <cnf>application/xml</cnf>
    <con>
        &lt;obj&gt;
            &lt;str name=&quot;type&quot; val=&quot;Sensor&quot;/&gt;
            &lt;str name=&quot;Category&quot; val=&quot;Category&quot;/&gt;
            &lt;str name=&quot;Unit&quot; val=&quot;Lux&quot;/&gt;
            &lt;str name=&quot;Model&quot; val=&quot;1142_0&quot;/&gt;
            &lt;str name=&quot;Location&quot; val=&quot;Home&quot;/&gt;
            &lt;str name=&quot;Manufacturer&quot; val=&quot;PHIDGETS&quot;/&gt;
            &lt;str name=&quot;Consumption Max&quot; val=&quot;27 mA&quot;/&gt;
            &lt;str name=&quot;Voltage Min&quot; val=&quot;4.8 V DC&quot;/&gt;
            &lt;str name=&quot;Voltage Max&quot; val=&quot;5.3 V DC&quot;/&gt;
            &lt;str name=&quot;Operating Temperature Max&quot; val=&quot;0
C&quot;/&gt;
            &lt;str name=&quot;Operating Temperature Min Max&quot; val=&quot;70
C&quot;/&gt;

        &lt;/obj&gt;
    </con>
</m2m:cin>
```

Figure 12: Script in XML for instance creation

POST http://127.0.0.1:8080/~/mn-cse/mn-name/SmartMeter/DATA Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL XML Beautify

```
<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/protocols">
    <cnf>application/xml</cnf>
    <con>
        &lt;obj&gt;
            &lt;str name=&quot;Category&quot; val=&quot;Light&quot;/&gt;
            &lt;str name=&quot;Data&quot; val=&quot;300&quot;/&gt;
            &lt;str name=&quot;Unit&quot; val=&quot;Lux&quot;/&gt;
            &lt;str name=&quot;Location&quot; val=&quot;Home&quot;/&gt;
        &lt;/obj&gt;
    </con>
</m2m:cin>
```

We can now see the resource tree for the MN-CSE with only the SmartMeter AE displayed:

OM2M CSE Resource Tree

http://127.0.0.1:8080/~/mn-cse/cin-800813701

- mn-name
  - acp_admin
  - ACP_Test
  - mon_acp
  - LuminositySensor
  - TemperatureSensor
  - SmartMeter
    - DESCRIPTOR
      - cin_5150100
    - DATA
      - cin_800813701
  - in-name

Attribute	Value
rn	cin_800813701
ty	4
ri	/mn-cse/cin-800813701
pi	/mn-cse/cnt-958031345
ct	20211012T180842
lt	20211012T180842
st	0
cnf	application/xml
cs	209
con	Attribute: Value Category: Light Data: 300 Unit: Lux Location: Home

Figure 13: OM2M ressource tree with the Smartmeter instances and containers

And with all the AEs displayed:

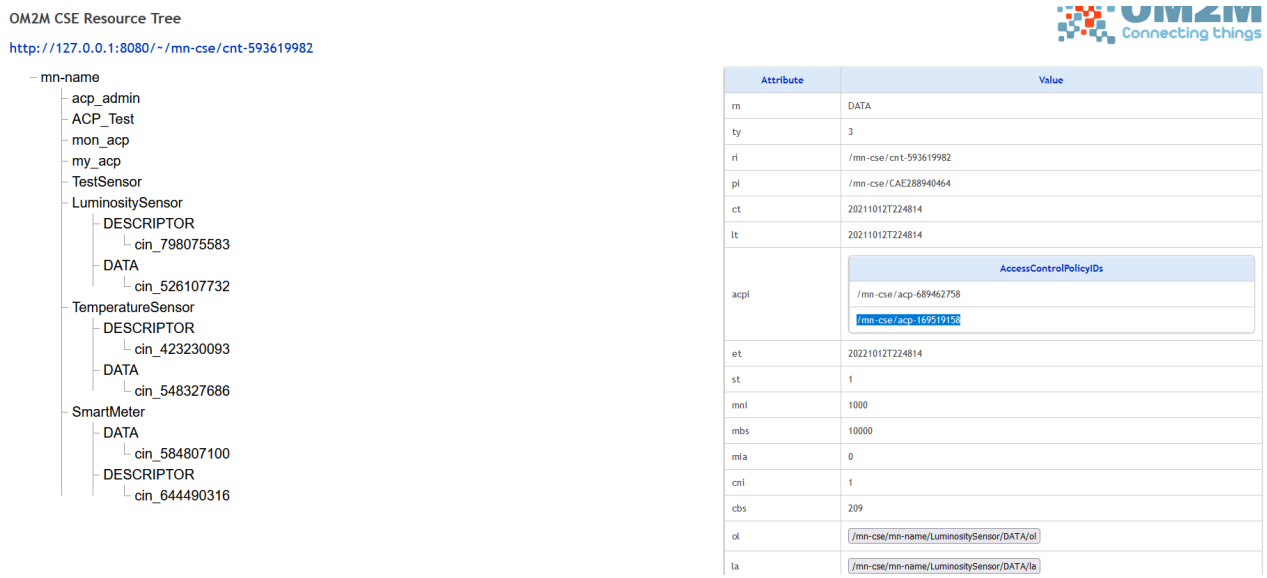

Figure 14: OM2M ressource tree with all the AE , their instances and containers

After all that, we started the monitoring application. This application listens on the localhost IP address on port 1400.

```
C:\Users\nguye\Documents>java -jar monitor.jar
Starting server..
The server is now listening on
Port: 1400
Context: /monitor
```

Figure 15: Monitor console for the java server

We can now create a new AE (type 1)with a request reachability attribute (rr) at true and a point of access (poa) with the url of the monitor. The request is shown in the next figure:

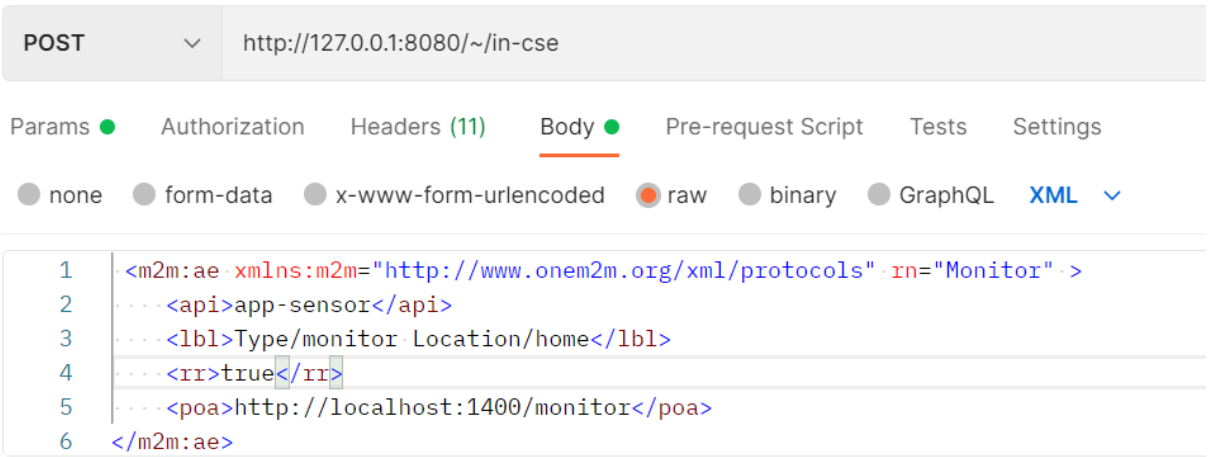

Figure 16: http POST request for the AE (Monitor) creation

Attribute	Value
rn	Monitor
ty	2
ri	/in-cse/CAE232149347
pi	/in-cse
ct	20211012T231822
lt	20211012T231822
lbl	• Type/monitor • Location/home
acpi	AccessControlPolicyIDs /in-cse/acp-836641350
et	20221012T231822
api	app-sensor
aei	CAE232149347
poa	Point Of Access http://localhost:1400/monitor
rr	true

Figure 17: Attributes of the Monitor AE in the OM2M ressource tree

Finally, we created subscriptions to allow the monitor to receive electricity, luminosity and temperature data when new values are stored in the platform. To do this, we made subscription resources in each DATA container. In the following figures, we have an example of the requests:

POST http://127.0.0.1:8080/~/mn-cse/mn-name/TemperatureSensor/DATA Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

Headers 9 hidden

	KEY	VALUE	DESCRIPTIC	ooo	Bulk Edit	Presets
☑	X-M2M-Origin	admin:admin				
☑	Content-Type	application/xml;ty=23				

Figure 18: http POST request for the subscription of the Data in TemperatureSensor

POST http://127.0.0.1:8080/~/mn-cse/mn-name/TemperatureSensor/DATA

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL XML

```
1 <m2m:sub xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="SUB_TEMPERATURE">
2     <nu>http://localhost:1400/monitor</nu>
3     <nct>2</nct>
4 </m2m:sub>
```

Figure 19: Script in XML for the subscription of the Data in TemperatureSensor

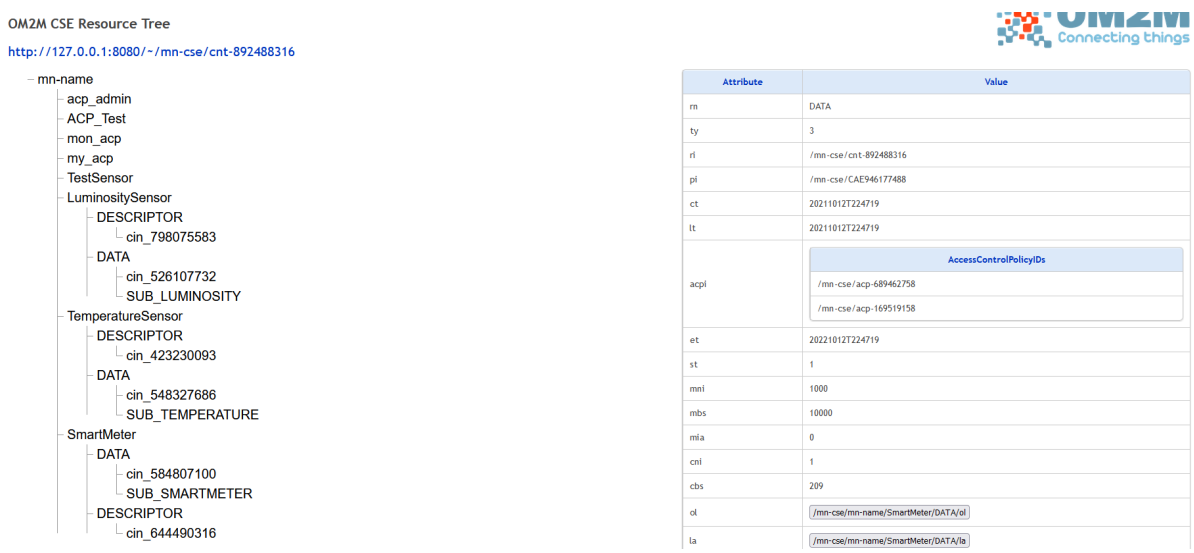

Figure 20: OM2M ressource tree with all the subscriptions to the Data of the AE

When the containers receive new data, the monitor is notified:

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

Received notification:
<?xml version="1.0" encoding="UTF-8"?>
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:hd="http://www.onem2m.org/xml/protocols/homedomain">
   <nev>
      <rep>
         <m2m:cin rn="cin_283193280">
            <ty>4</ty>
            <ri>/mn-cse/cin-283193280</ri>
            <pi>/mn-cse/cnt-593619982</pi>
            <ct>20211013T121449</ct>
            <lt>20211013T121449</lt>
            <st>0</st>
            <cnf>application/xml</cnf>
            <cs>209</cs>
            <con>
        &lt;obj>
            &lt;str name=&quot;Category&quot; val=&quot;Light&quot;/>
            &lt;str name=&quot;Data&quot; val=&quot;165&quot;/>
            &lt;str name=&quot;Unit&quot; val=&quot;Lux&quot;/>
            &lt;str name=&quot;Location&quot; val=&quot;Home&quot;/>
        &lt;/obj>
    </con>
         </m2m:cin>
      </rep>
      <rss>1</rss>
   </nev>
   <sud>false</sud>
   <sur>/mn-cse/sub-13495376</sur>
</m2m:sgn>
```

Figure 21: Notification of a new data in the monitor

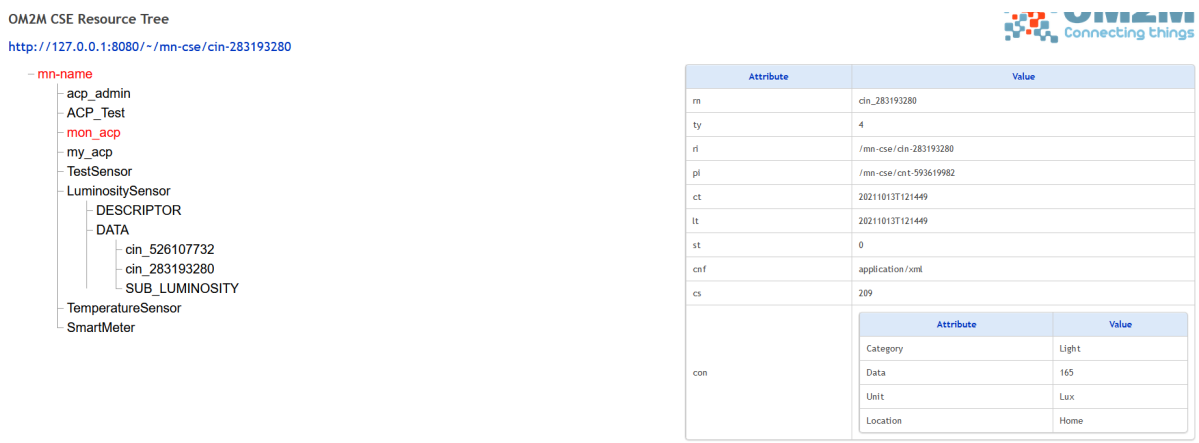

Figure 22: Creation of a new data in the OM2M ressource tree

With these information, the monitor can now use the data as the user wants it to.