Lab 3: OM2M/REST API & Node-RED

物聯網技術與應用(英) IoT/M2M Technologies and Applications

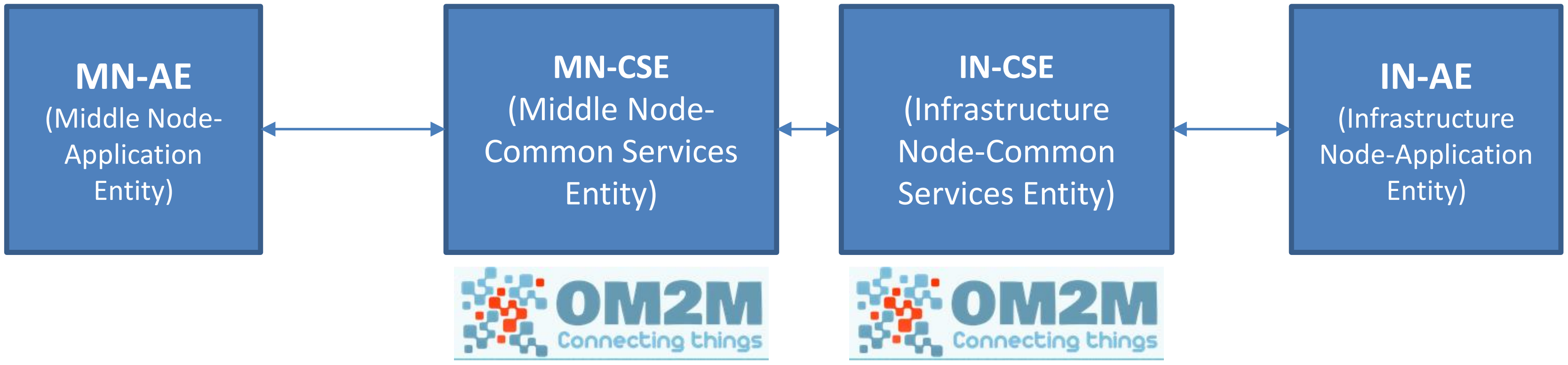
國立交通大學資訊工程系
Department of Computer Science
National Chiao Tung University

October 26, 2018

Outline

- High Level Architecture.
- RESTful Operations on the Resource Tree using PostMan.
  - Creating Application, Descriptor Container, Data Container, Descriptor ContentInstance, Data ContentInstance. *(Checkpoint 1)*
  - Creating and Testing a Subscription. *(Checkpoint 2)*
- RESTful Operations on the Resource Tree using Node-RED.
  - Creating a Middle Node-Application Entity with Node-RED Part I. *(Checkpoint 3)*
  - Creating a Middle Node-Application Entity with Node-RED Part II. *(Checkpoint 4)*
  - Extending the Middle Node-Application Entity with HTTP Server Capabilities. *(Checkpoint 5)*
  - Creating a subscription resource to a container in Middle Node-Application Entity using Node-RED. *(Checkpoint 6)*

High Level Architecture

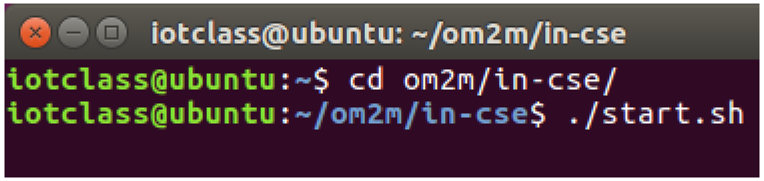

MN-AE (Middle Node-Application Entity)
MN-CSE (Middle Node-Common Services Entity)
IN-CSE (Infrastructure Node-Common Services Entity)
IN-AE (Infrastructure Node-Application Entity)
OM2M Connecting things
OM2M Connecting things

Start OM2M IN-CSE

- Open a Terminal window in your VM
- Enter the following commands:

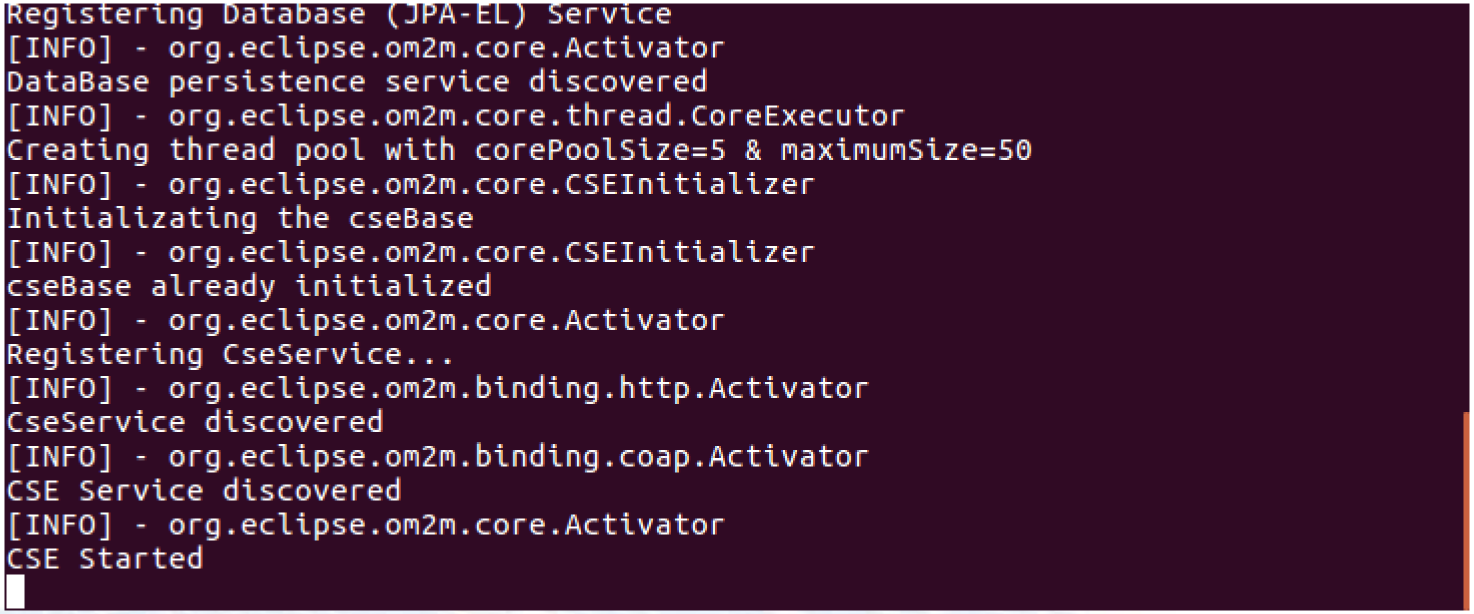
  $ cd om2m/in-cse

  $ ./start.sh



```
iotclass@ubuntu: ~/om2m/in-cse
iotclass@ubuntu:~$ cd om2m/in-cse/
iotclass@ubuntu:~/om2m/in-cse$ ./start.sh
```

- After starting it successfully, you will see “CSE Started” in your terminal.



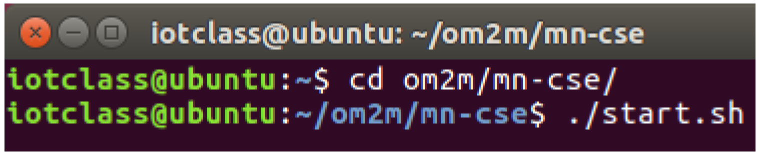
```
Registering Database (JPA-EL) Service
[INFO] - org.eclipse.om2m.core.Activator
DataBase persistence service discovered
[INFO] - org.eclipse.om2m.core.thread.CoreExecutor
Creating thread pool with corePoolSize=5 & maximumSize=50
[INFO] - org.eclipse.om2m.core.CSEInitializer
Initializating the cseBase
[INFO] - org.eclipse.om2m.core.CSEInitializer
cseBase already initialized
[INFO] - org.eclipse.om2m.core.Activator
Registering CseService...
[INFO] - org.eclipse.om2m.binding.http.Activator
CseService discovered
[INFO] - org.eclipse.om2m.binding.coap.Activator
CSE Service discovered
[INFO] - org.eclipse.om2m.core.Activator
CSE Started
```

Start OM2M MN-CSE

- **Open another** Terminal window in your VM.
- Enter the following commands:

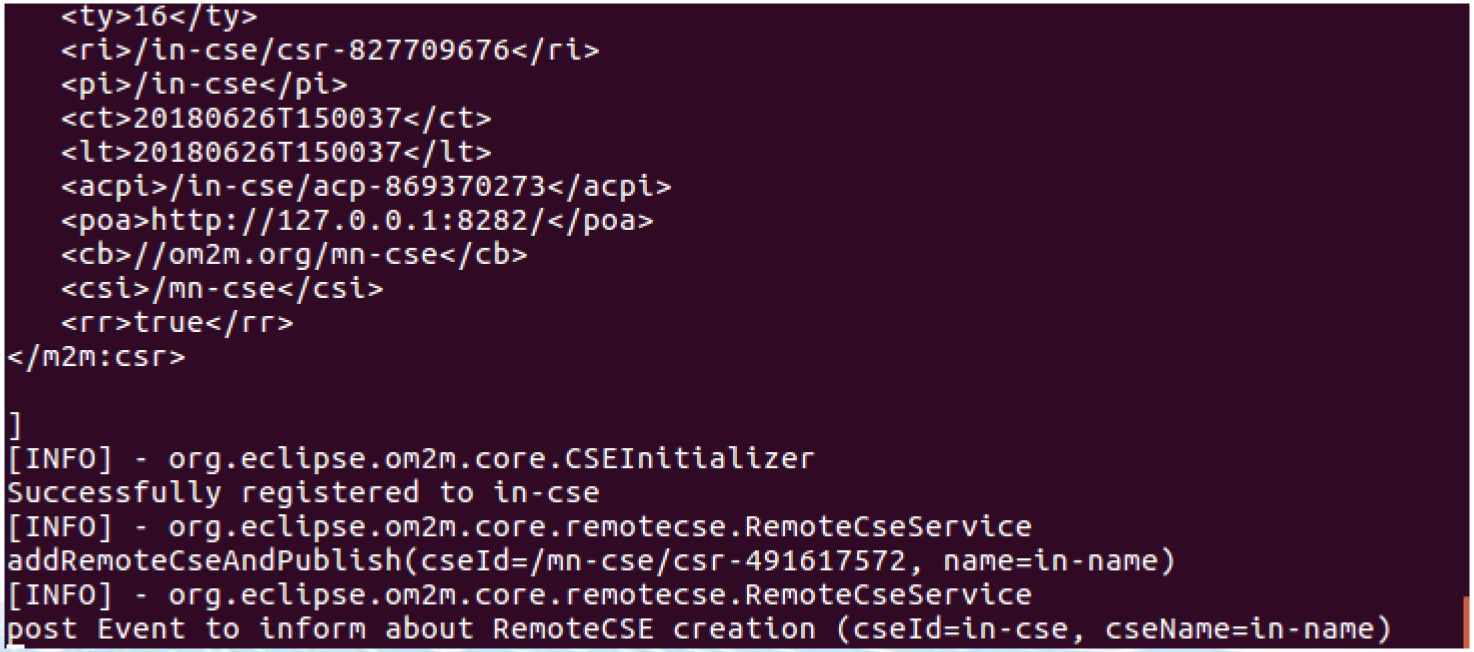
  $ cd om2m/mn-cse

  $ ./start.sh



```
iotclass@ubuntu: ~/om2m/mn-cse
iotclass@ubuntu:~$ cd om2m/mn-cse/
iotclass@ubuntu:~/om2m/mn-cse$ ./start.sh
```

- After starting it successfully, you will see “post Event to inform about RemoteCSE creation …” in your terminal.



```
    <ty>16</ty>
    <ri>/in-cse/csr-827709676</ri>
    <pi>/in-cse</pi>
    <ct>20180626T150037</ct>
    <lt>20180626T150037</lt>
    <acpi>/in-cse/acp-869370273</acpi>
    <poa>http://127.0.0.1:8282/</poa>
    <cb>//om2m.org/mn-cse</cb>
    <csi>/mn-cse</csi>
    <rr>true</rr>
</m2m:csr>

]
[INFO] - org.eclipse.om2m.core.CSEInitializer
Successfully registered to in-cse
[INFO] - org.eclipse.om2m.core.remotecse.RemoteCseService
addRemoteCseAndPublish(cseId=/mn-cse/csr-491617572, name=in-name)
[INFO] - org.eclipse.om2m.core.remotecse.RemoteCseService
post Event to inform about RemoteCSE creation (cseId=in-cse, cseName=in-name)
```

RESTful Operations on the Resource Tree using PostMan

Retrieve a resource:

URL	http://127.0.0.1:8282/~/mn-cse
Method	GET
Header	X-M2M-Origin : admin:admin
Body	(empty)

HTTP Response:

Field	Value
Status	200 OK

Body	`<?xml version="1.0" encoding="UTF-8"?>`
	`<m2m:cb xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="mn-name">`
	`    <ty>5</ty>`
	`    <ri>/mn-cse</ri>`
	`    <ct>20160628T124737</ct>`
	`    <lt>20160628T124737</lt>`
	`    <acpi>/mn-cse/acp-832322075</acpi>`
	`    <cst>1</cst>`
	`    <csi>mn-cse</csi>`
	`    <srt>1 2 3 4 5 9 14 15 16 17 23</srt>`
	`    <poa>http://127.0.0.1:8282/</poa>`
	`</m2m:cb>`

RESTful Operations on the Resource Tree using PostMan

Create a "MY_SENSOR" Application (1/2)

- Send an HTTP POST request with the following parameters to create a "MY_SENSOR" application (MN-AE) on the gateway.

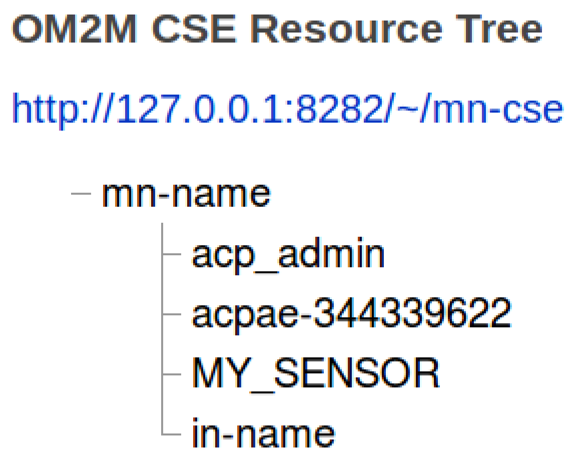
URL	http://127.0.0.1:8282/~/mn-cse
Method	POST
Header	X-M2M-Origin: admin:admin Content-Type: application/xml;ty=2

Body	`<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="MY_SENSOR" >`
	    `<api>app-sensor</api>`
	    `<lbl>Type/sensor Category/temperature Location/home</lbl>`
	    `<rr>false</rr>`
	`</m2m:ae>`

RESTful Operations on the Resource Tree using PostMan

Create a "MY_SENSOR" Application(2/2)

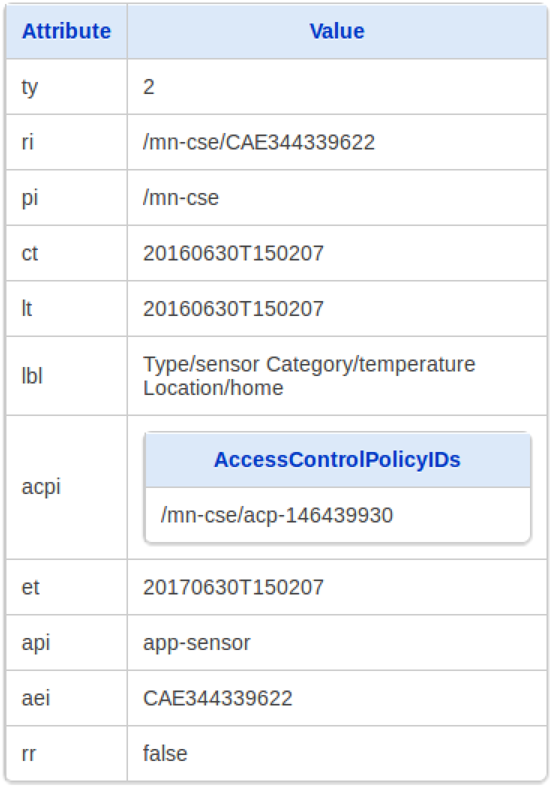
- You will get Status 201 Created in Postman
- Check the created application on the OM2M Resource Tree Tool.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse

```
– mn-name
    ├ acp_admin
    ├ acpae-344339622
    ├ MY_SENSOR
    └ in-name
```



Attribute	Value
ty	2
ri	/mn-cse/CAE344339622
pi	/mn-cse
ct	20160630T150207
lt	20160630T150207
lbl	Type/sensor Category/temperature Location/home
acpi	AccessControlPolicyIDs: /mn-cse/acp-146439930
et	20170630T150207
api	app-sensor
aei	CAE344339622
rr	false

RESTful Operations on the Resource Tree using PostMan

Discover Resources based on their labels (1/2)

- Discover available resources based on their search strings using the discovery resource.
- The **fu** parameter (stands for "Filter Usage") indicates that it is a discovery request.
- The **lbl** ("Label") indicates the specific label you want to search for.

URL	http://127.0.0.1:8282/~/mn-cse?fu=1&lbl=Type/sensor
Method	GET
Header	X-M2M-Origin: admin:admin Content-Type: application/xml
Body	(empty)

RESTful Operations on the Resource Tree using PostMan

Discover Resources based on their labels (2/2)

You will get Status 200 in Postman

- Check the retrieved XML data. It shows the URI of the "MY_SENSOR" AE you just created.

Field	Value
Status	200 OK
Body	<?xml version="1.0" encoding="UTF-8"?> <m2m:uril xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:hd="http://www.onem2m.org/xml/protocols/homedomain"> /mn-cse/mn-name/MY_SENSOR </m2m:uril>

RESTful Operations on the Resource Tree using PostMan

Create a DESCRIPTOR Container(1/2)

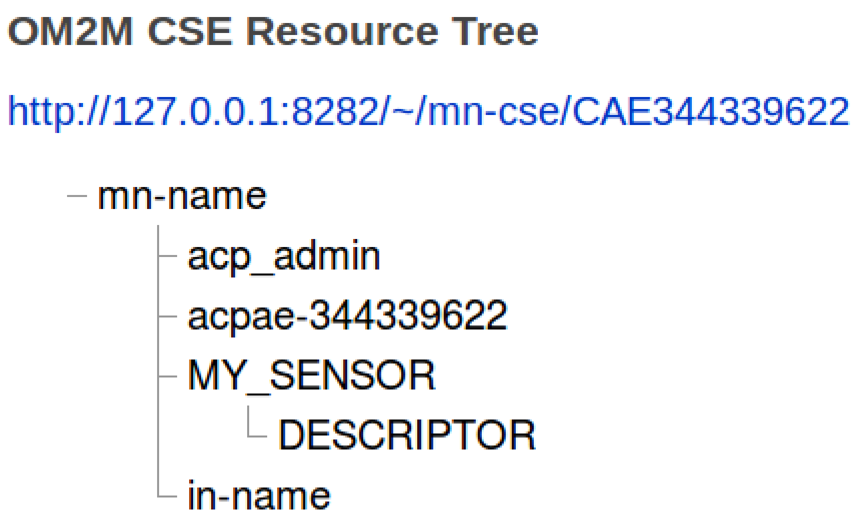
- Send an HTTP POST request with the following parameters to create a "DESCRIPTOR" container resource under the "MY_SENSOR" application.

URL	http://127.0.0.1:8282/~/mn-cse/mn-name/MY_SENSOR
Method	POST
Header	X-M2M-Origin: admin:admin Content-Type: application/xml;ty=3
Body	<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="DESCRIPTOR"> </m2m:cnt>

RESTful Operations on the Resource Tree using PostMan

Create a DESCRIPTOR Container (2/2)

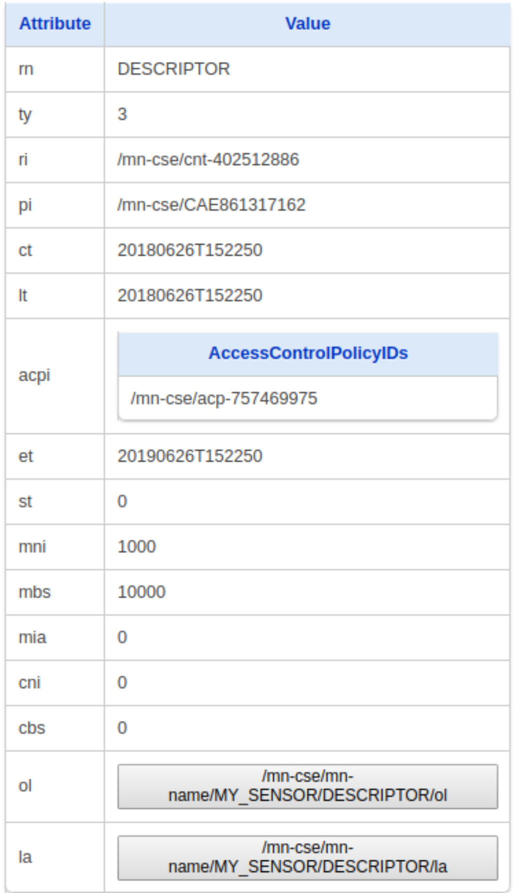
- Check the created container on the OM2M Resource Tree Tool.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse/CAE344339622

```
─ mn-name
    ├ acp_admin
    ├ acpae-344339622
    ├ MY_SENSOR
    │   └ DESCRIPTOR
    └ in-name
```



Attribute	Value
rn	DESCRIPTOR
ty	3
ri	/mn-cse/cnt-402512886
pi	/mn-cse/CAE861317162
ct	20180626T152250
lt	20180626T152250
acpi	AccessControlPolicyIDs: /mn-cse/acp-757469975
et	20190626T152250
st	0
mni	1000
mbs	10000
mia	0
cni	0
cbs	0
ol	/mn-cse/mn-name/MY_SENSOR/DESCRIPTOR/ol
la	/mn-cse/mn-name/MY_SENSOR/DESCRIPTOR/la

RESTful Operations on the Resource Tree using PostMan

Create a DESCRIPTOR ContentInstance (1/2)

- Send an HTTP POST request with the following parameters to create a descriptor content instance resource under the "DESCRIPTOR" container.

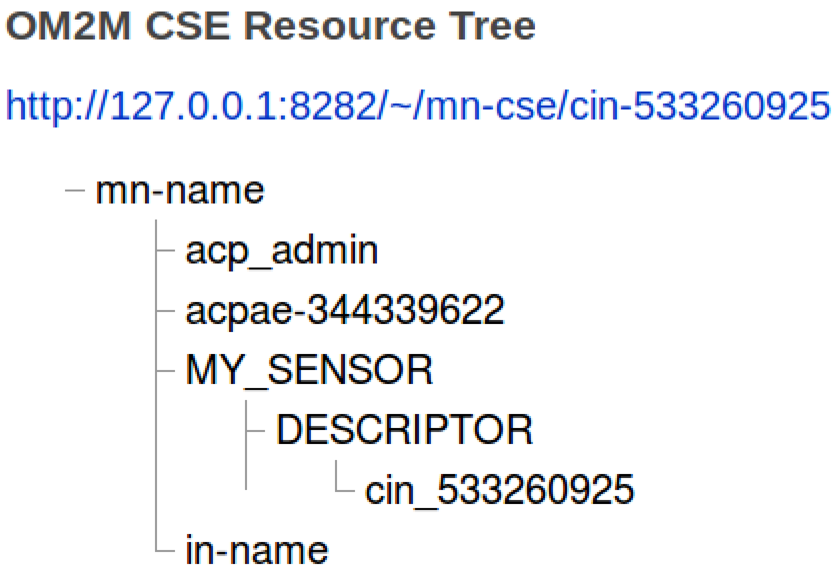
URL	http://127.0.0.1:8282/~/mn-cse/mn-name/MY_SENSOR/DESCRIPTOR
Method	POST
Header	X-M2M-Origin: admin:admin Content-Type: application/xml;ty=4

Body	`<om2m:cin xmlns:om2m="http://www.onem2m.org/xml/protocols">`
	    `<cnf>message</cnf>`
	    `<con>`
	        `&lt;obj&gt;`
	            `&lt;str name=&quot;type&quot; val=&quot;Temperature_Sensor&quot;/&gt;`
	            `&lt;str name=&quot;location&quot; val=&quot;Home&quot;/&gt;`
	            `&lt;str name=&quot;appId&quot; val=&quot;MY_SENSOR&quot;/&gt;`
	            `&lt;op name=&quot;getValue&quot; href=&quot;/mn-cse/mn-name/MY_SENSOR/DATA/la&quot;`
	         `in=&quot;obix:Nil&quot; out=&quot;obix:Nil&quot; is=&quot;retrieve&quot;/&gt;`
	        `&lt;/obj&gt;`
	    `</con>`
	`</om2m:cin>`

RESTful Operations on the Resource Tree using PostMan

Create a DESCRIPTOR ContentInstance (2/2)

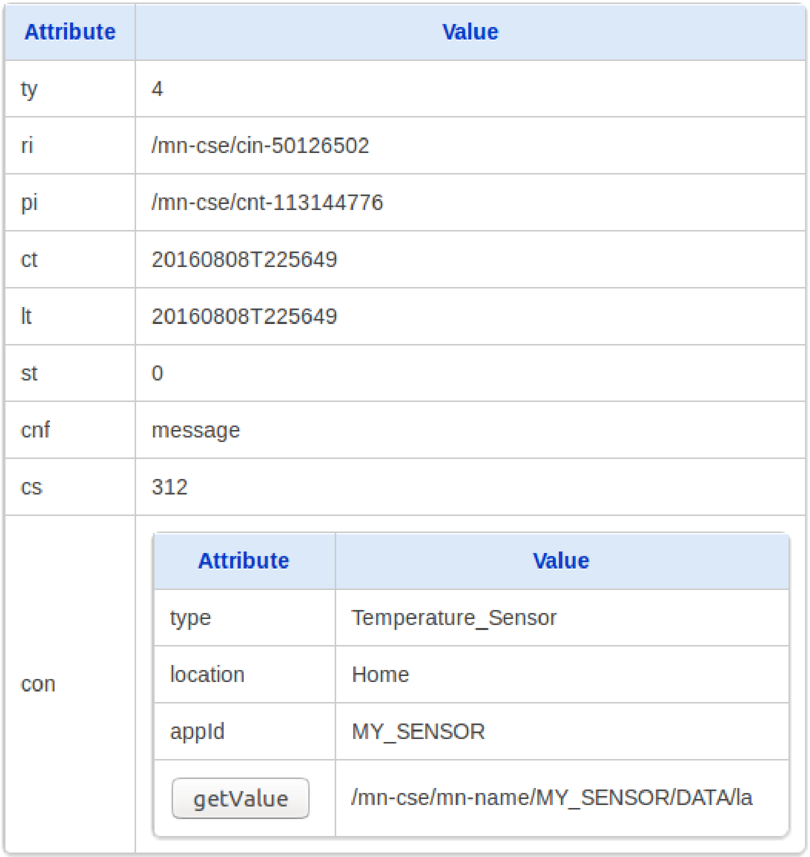
- Check the created content instance on the OM2M Resource Tree Tool.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse/cin-533260925

```
─ mn-name
    ├ acp_admin
    ├ acpae-344339622
    ├ MY_SENSOR
    │   ├ DESCRIPTOR
    │   │   └ cin_533260925
    └ in-name
```



Attribute	Value
ty	4
ri	/mn-cse/cin-50126502
pi	/mn-cse/cnt-113144776
ct	20160808T225649
lt	20160808T225649
st	0
cnf	message
cs	312
con	(nested table below)

Attribute	Value
type	Temperature_Sensor
location	Home
appId	MY_SENSOR
getValue	/mn-cse/mn-name/MY_SENSOR/DATA/la

RESTful Operations on the Resource Tree using PostMan

Create a DATA Container (1/2)

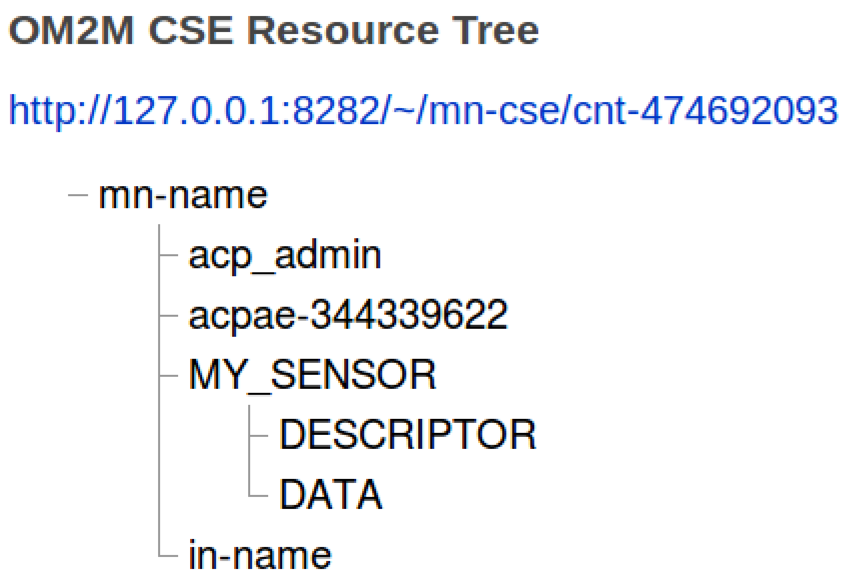
- Send an HTTP POST request with the following parameters to create a "DATA" container resource under the "MY_SENSOR" application.

URL	http://127.0.0.1:8282/~/mn-cse/mn-name/MY_SENSOR
Method	POST
Header	X-M2M-Origin: admin:admin Content-Type: application/xml;ty=3
Body	<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="DATA"> </m2m:cnt>

RESTful Operations on the Resource Tree using PostMan

Create a DATA Container (2/2)

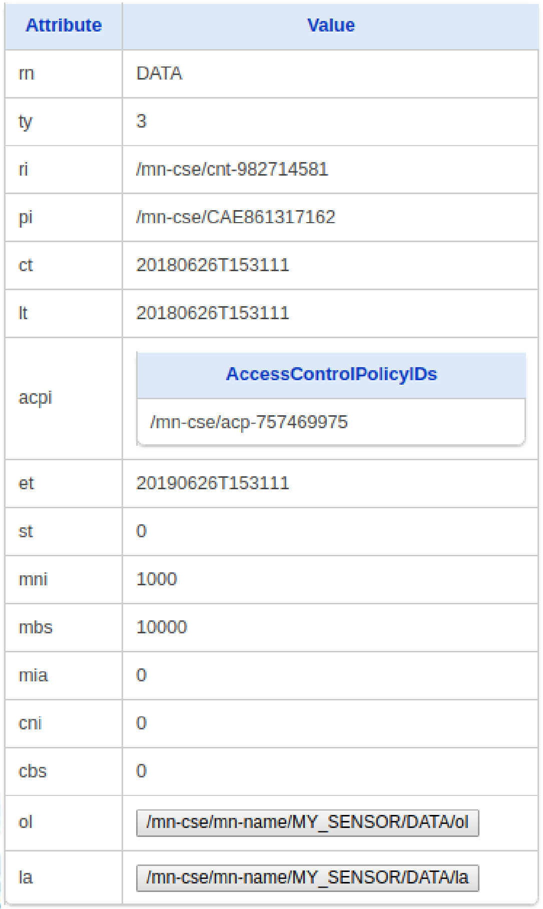
- Check the created container on the OM2M Resource Tree Tool.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse/cnt-474692093

```
− mn-name
   ├ acp_admin
   ├ acpae-344339622
   ├ MY_SENSOR
   │   ├ DESCRIPTOR
   │   └ DATA
   └ in-name
```



Attribute	Value
rn	DATA
ty	3
ri	/mn-cse/cnt-982714581
pi	/mn-cse/CAE861317162
ct	20180626T153111
lt	20180626T153111
acpi	AccessControlPolicyIDs /mn-cse/acp-757469975
et	20190626T153111
st	0
mni	1000
mbs	10000
mia	0
cni	0
cbs	0
ol	/mn-cse/mn-name/MY_SENSOR/DATA/ol
la	/mn-cse/mn-name/MY_SENSOR/DATA/la

RESTful Operations on the Resource Tree using PostMan

Create a DATA ContentInstance (1/2)

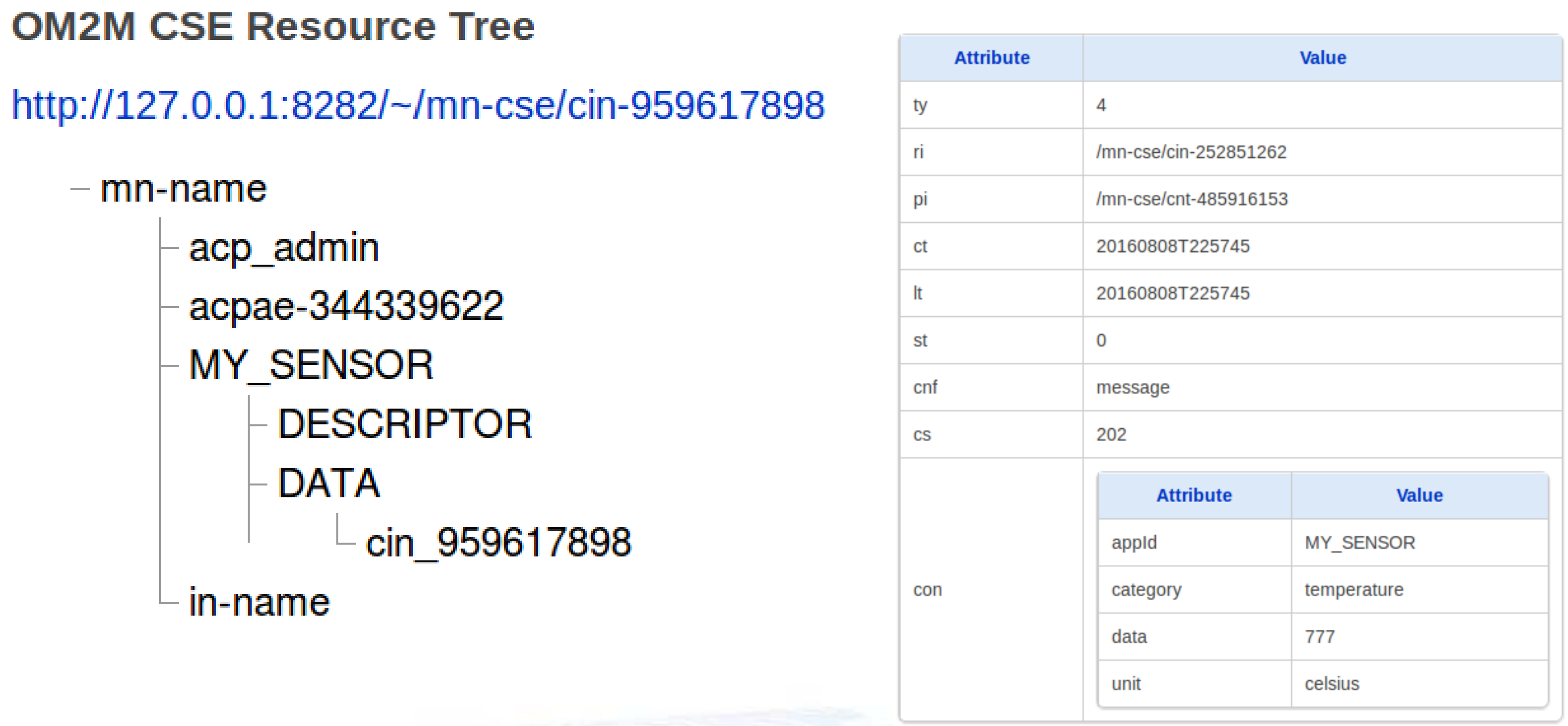
- Send an HTTP POST request with the following parameters to create a data “ContentInstance” resource under the "DATA" container.

URL	http://127.0.0.1:8282/~/mn-cse/mn-name/MY_SENSOR/DATA
Method	POST
Header	X-M2M-Origin: admin:admin Content-Type: application/xml;ty=4
Body	`<om2m:cin xmlns:om2m="http://www.onem2m.org/xml/protocols"> <cnf>message</cnf> <con> &lt;obj&gt; &lt;str name=&quot;appId&quot; val=&quot;MY_SENSOR&quot;/&gt; &lt;str name=&quot;category&quot; val=&quot;temperature &quot;/&gt; &lt;int name=&quot;data&quot; val=&quot;27&quot;/&gt; &lt;int name=&quot;unit&quot; val=&quot;celsius&quot;/&gt; &lt;/obj&gt; </con> </om2m:cin>`

RESTful Operations on the Resource Tree using PostMan

Create a DATA ContentInstance (2/2)

- Check the created ContentInstance on the OM2M Resource Tree Tool.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse/cin-959617898

```
─ mn-name
    ├ acp_admin
    ├ acpae-344339622
    ├ MY_SENSOR
    │   ├ DESCRIPTOR
    │   ├ DATA
    │   │   └ cin_959617898
    └ in-name
```

Attribute	Value
ty	4
ri	/mn-cse/cin-252851262
pi	/mn-cse/cnt-485916153
ct	20160808T225745
lt	20160808T225745
st	0
cnf	message
cs	202
con	

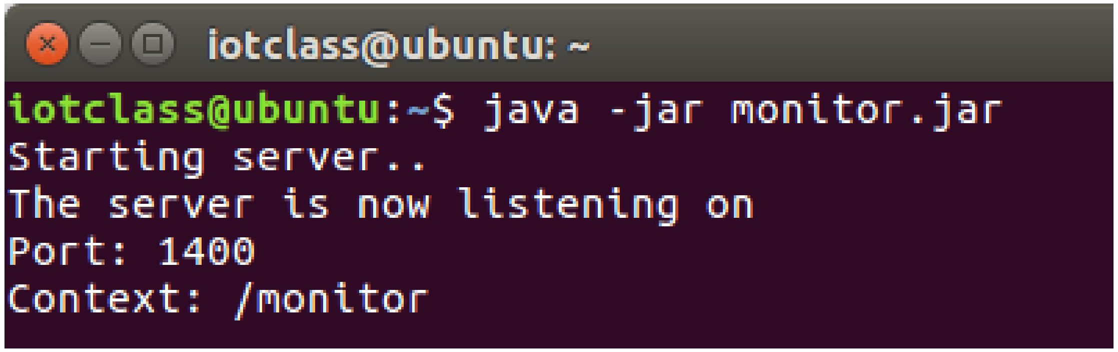
Attribute	Value
appId	MY_SENSOR
category	temperature
data	777
unit	celsius

CHECKPOINT 1!

RESTful Operations on the Resource Tree using PostMan

Subscribe to MY_SENSOR Data(1/5)

- Open a terminal window.
- Move to monitor folder : /home/iotclass
- Start the Monitor server using the following command:
  - *java -jar monitor.jar*
- Monitor is a Web Application that listens for HTTP Post requests at port=1400 and context=/monitor.



```
iotclass@ubuntu: ~
iotclass@ubuntu:~$ java -jar monitor.jar
Starting server..
The server is now listening on
Port: 1400
Context: /monitor
```

RESTful Operations on the Resource Tree using PostMan

Subscribe to MY_SENSOR Data (2/5)

- Send an HTTP POST request with the following parameters to create a “SUBSCRIPTION" resource for receiving asynchronous notifications from "MY_SENSOR" application when “MY_SENSOR” gets new data.
- The monitor server listening URL (**http://127.0.0.1:1400/monitor**) should be added on the “nu" tag of the subscription representation.

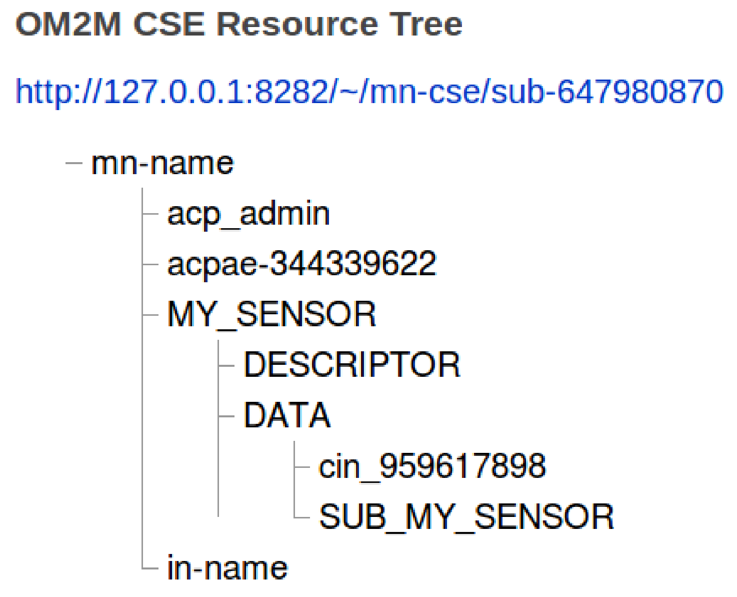
URL	http://127.0.0.1:8282/~/mn-cse/mn-name/MY_SENSOR/DATA
Method	POST
Header	X-M2M-Origin: admin:admin Content-Type: application/xml;ty=23

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

RESTful Operations on the Resource Tree using PostMan

Subscribe to MY_SENSOR Data (3/5)

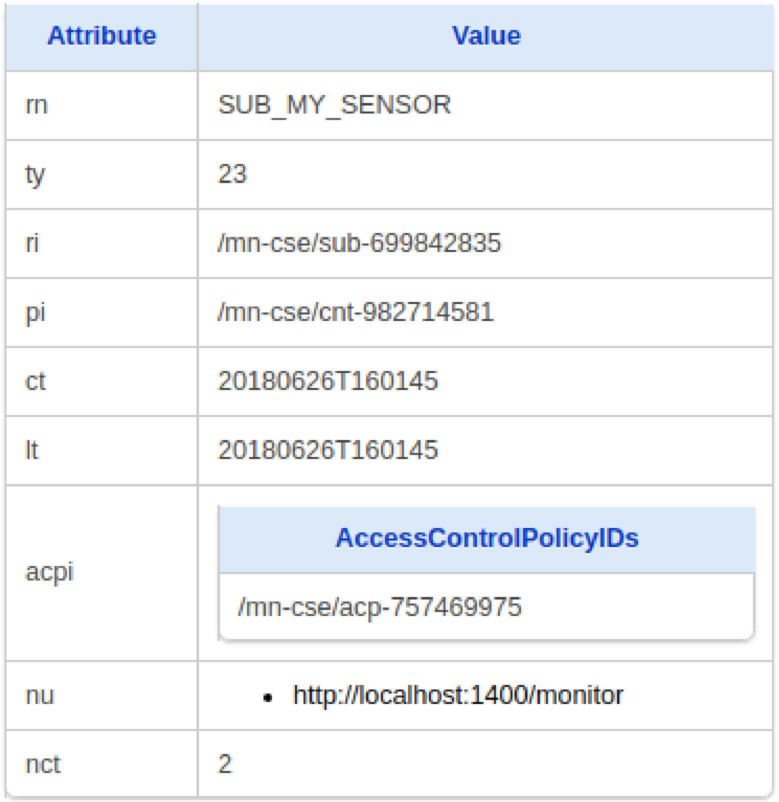
- Check the created subscription on the OM2M Resource Tree Tool.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse/sub-647980870

```
– mn-name
    acp_admin
    acpae-344339622
    MY_SENSOR
        DESCRIPTOR
        DATA
            cin_959617898
            SUB_MY_SENSOR
    in-name
```

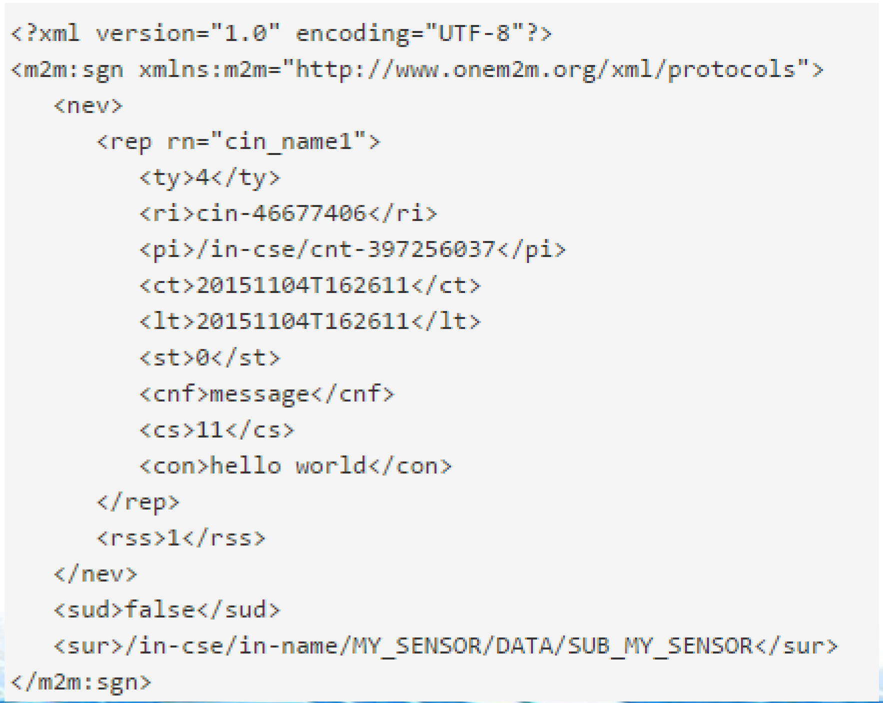


Attribute	Value
rn	SUB_MY_SENSOR
ty	23
ri	/mn-cse/sub-699842835
pi	/mn-cse/cnt-982714581
ct	20180626T160145
lt	20180626T160145
acpi	AccessControlPolicyIDs: /mn-cse/acp-757469975
nu	• http://localhost:1400/monitor
nct	2

RESTful Operations on the Resource Tree using PostMan

Subscribe to MY_SENSOR Data (4/5)

- Insert data on “MY_SENSOR” application (See slide 17). Inserting new data in the ContentInstance of DATA container will trigger a notification event.
- The new event is published to subscribers by the Gateway via HTTP POST Notification (an example of an XML Object returned to subscribers is shown below).

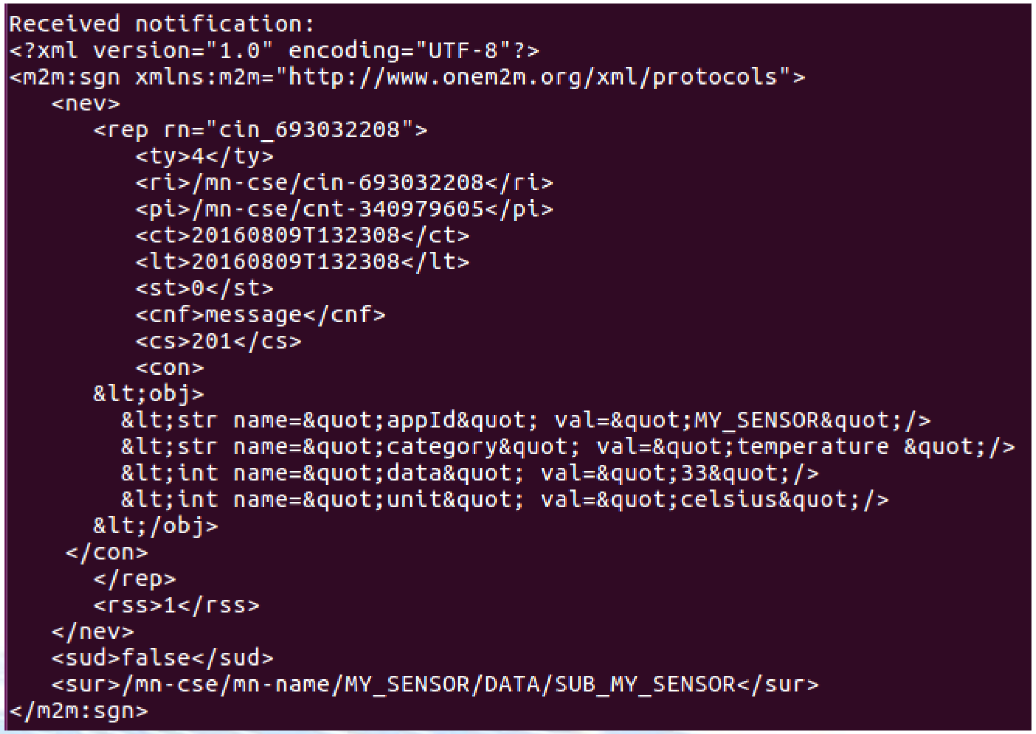


```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
   <nev>
      <rep rn="cin_name1">
         <ty>4</ty>
         <ri>cin-46677406</ri>
         <pi>/in-cse/cnt-397256037</pi>
         <ct>20151104T162611</ct>
         <lt>20151104T162611</lt>
         <st>0</st>
         <cnf>message</cnf>
         <cs>11</cs>
         <con>hello world</con>
      </rep>
      <rss>1</rss>
   </nev>
   <sud>false</sud>
   <sur>/in-cse/in-name/MY_SENSOR/DATA/SUB_MY_SENSOR</sur>
</m2m:sgn>
```

RESTful Operations on the Resource Tree using PostMan

Subscribe to MY_SENSOR Data (5/5)

- The Monitor receives a "Notify" resource including the new "ContentInstance".
- Remember that the "Notify" resource has a generic structure to support notifications for other type of resources such as “AE", "Container", "Group", AccessControlPolicy, etc.



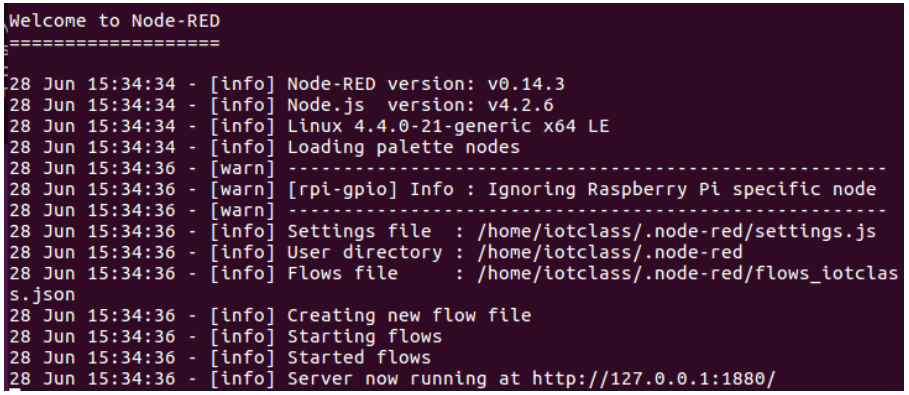
```
Received notification:
<?xml version="1.0" encoding="UTF-8"?>
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
   <nev>
      <rep rn="cin_693032208">
         <ty>4</ty>
         <ri>/mn-cse/cin-693032208</ri>
         <pi>/mn-cse/cnt-340979605</pi>
         <ct>20160809T132308</ct>
         <lt>20160809T132308</lt>
         <st>0</st>
         <cnf>message</cnf>
         <cs>201</cs>
         <con>
      &lt;obj>
        &lt;str name=&quot;appId&quot; val=&quot;MY_SENSOR&quot;/>
        &lt;str name=&quot;category&quot; val=&quot;temperature &quot;/>
        &lt;int name=&quot;data&quot; val=&quot;33&quot;/>
        &lt;int name=&quot;unit&quot; val=&quot;celsius&quot;/>
      &lt;/obj>
    </con>
      </rep>
      <rss>1</rss>
   </nev>
   <sud>false</sud>
   <sur>/mn-cse/mn-name/MY_SENSOR/DATA/SUB_MY_SENSOR</sur>
</m2m:sgn>
```

CHECKPOINT 2!

Creating a MN-AE with Node-RED Part I

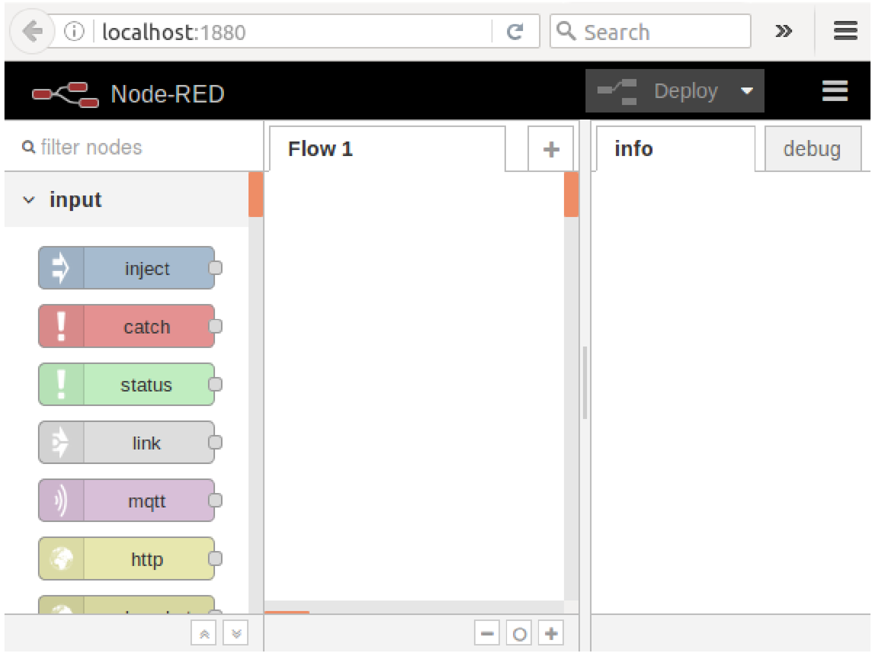
Start the Node-RED

- Open a new terminal and input the command:
  - $ node-red
- Open http://localhost:1880 in a browser.



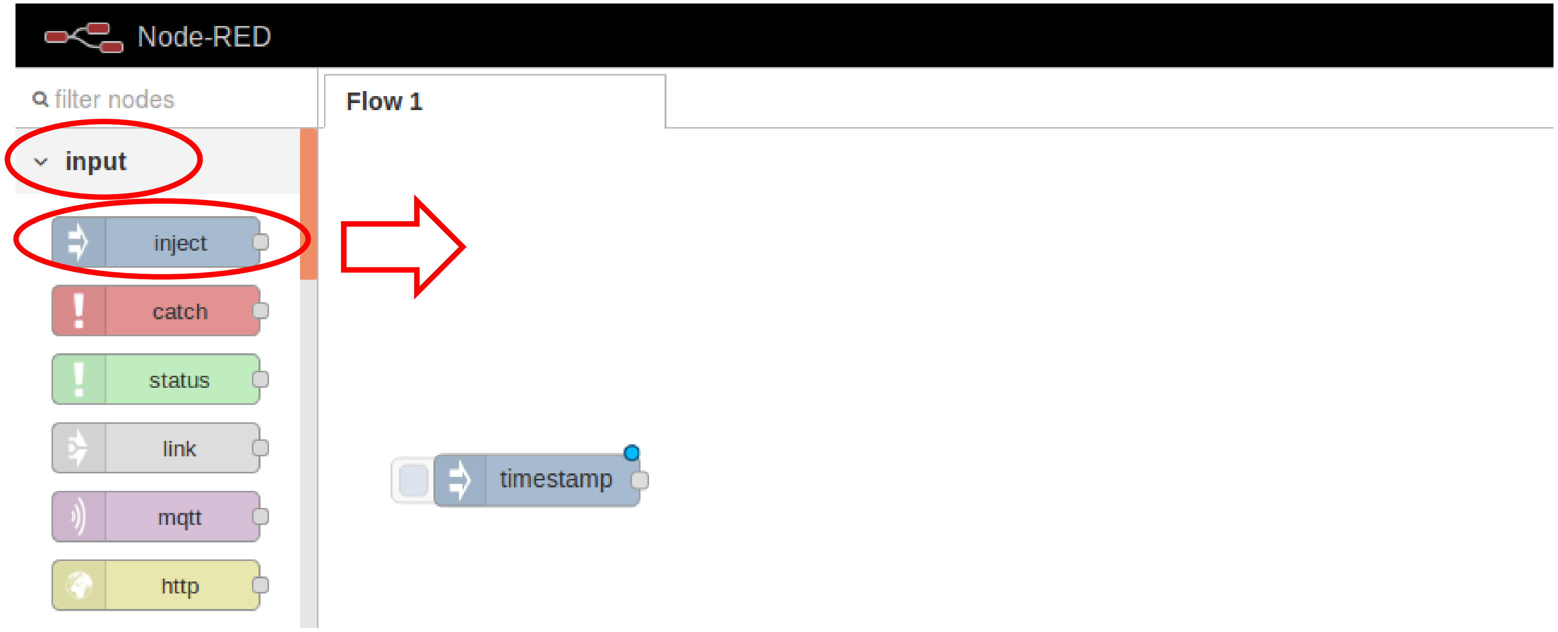
```
Welcome to Node-RED
===================

28 Jun 15:34:34 - [info] Node-RED version: v0.14.3
28 Jun 15:34:34 - [info] Node.js  version: v4.2.6
28 Jun 15:34:34 - [info] Linux 4.4.0-21-generic x64 LE
28 Jun 15:34:34 - [info] Loading palette nodes
28 Jun 15:34:36 - [warn] ------------------------------------------------------
28 Jun 15:34:36 - [warn] [rpi-gpio] Info : Ignoring Raspberry Pi specific node
28 Jun 15:34:36 - [warn] ------------------------------------------------------
28 Jun 15:34:36 - [info] Settings file  : /home/iotclass/.node-red/settings.js
28 Jun 15:34:36 - [info] User directory : /home/iotclass/.node-red
28 Jun 15:34:36 - [info] Flows file     : /home/iotclass/.node-red/flows_iotclas
s.json
28 Jun 15:34:36 - [info] Creating new flow file
28 Jun 15:34:36 - [info] Starting flows
28 Jun 15:34:36 - [info] Started flows
28 Jun 15:34:36 - [info] Server now running at http://127.0.0.1:1880/
```



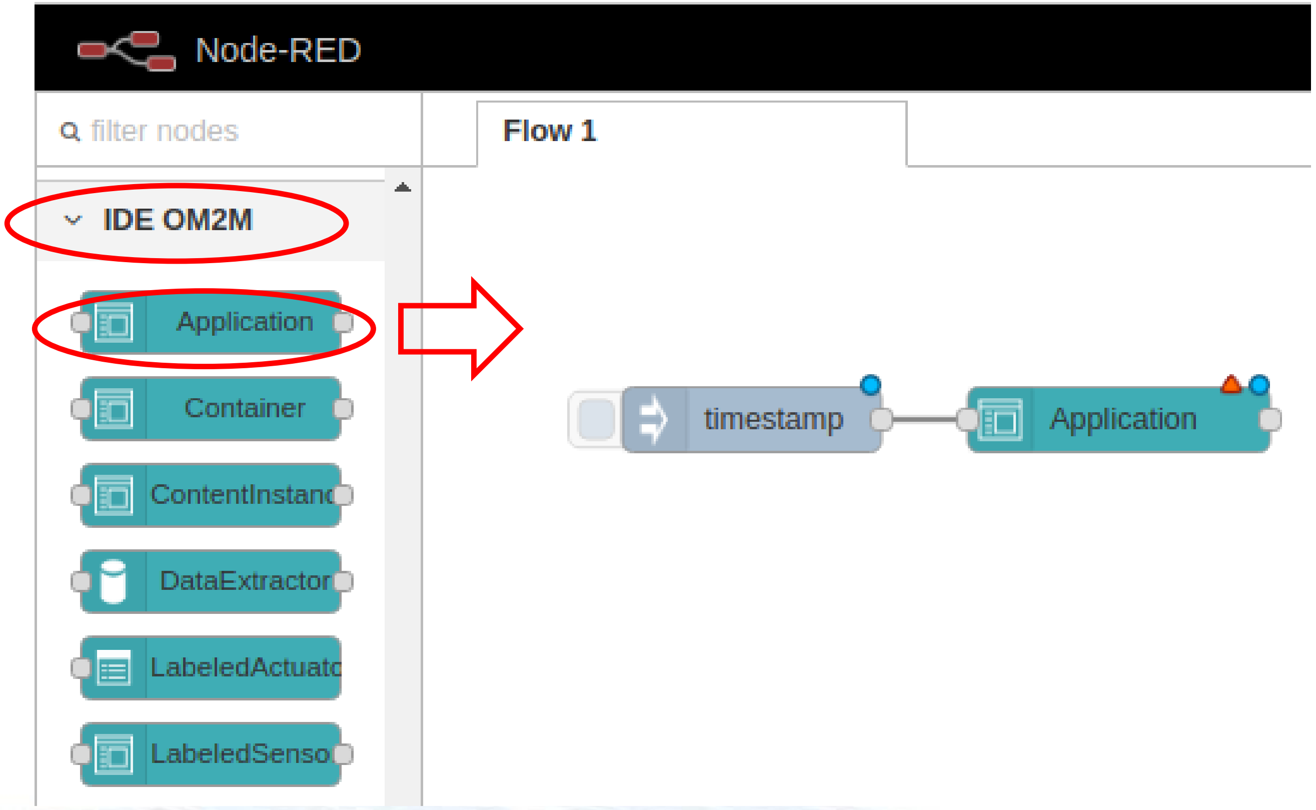
Creating a MN-AE with Node-RED Part I

1. From the "input" library, drag and drop an "Inject" object into the designer area.



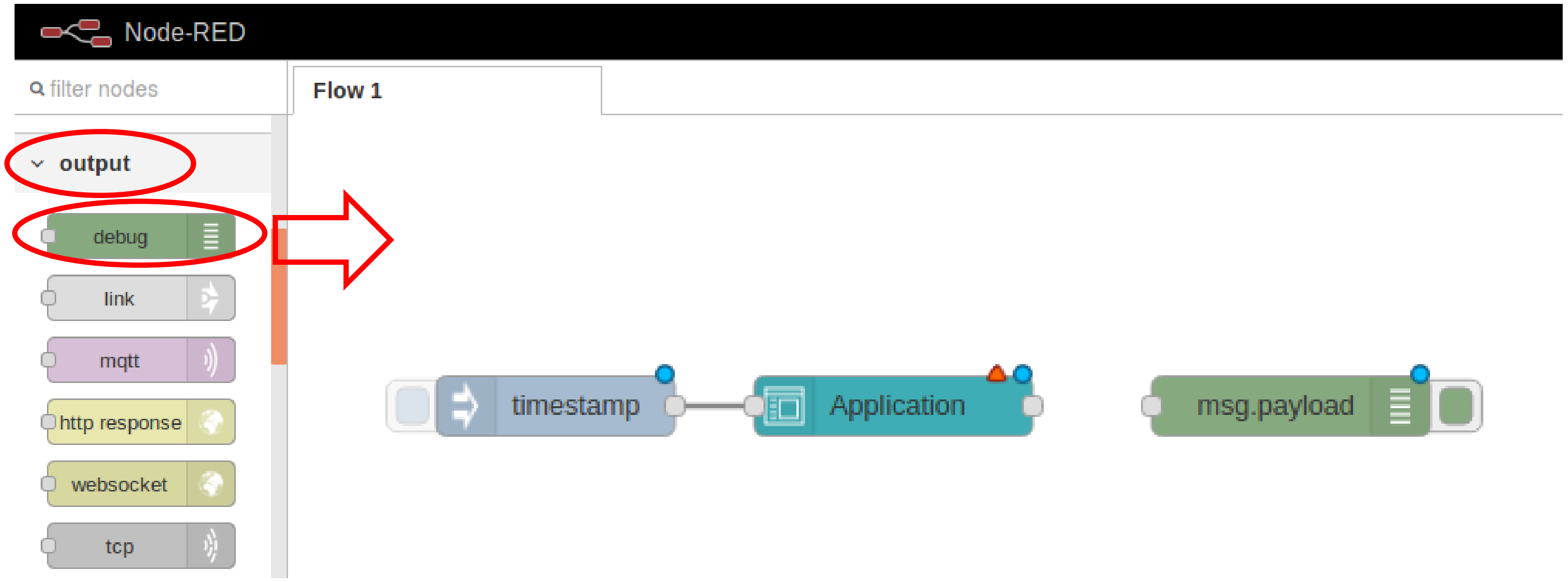
Creating a MN-AE with Node-RED Part I

2. From the “IDE OM2M” library, drag and drop an “Application” object into the designer area.



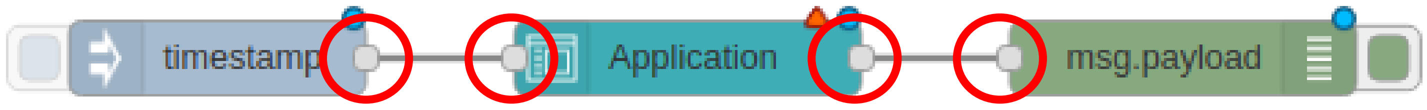
Creating a MN-AE with Node-RED Part I

3. From the "output" library, drag and drop a "debug" object into the designer area.



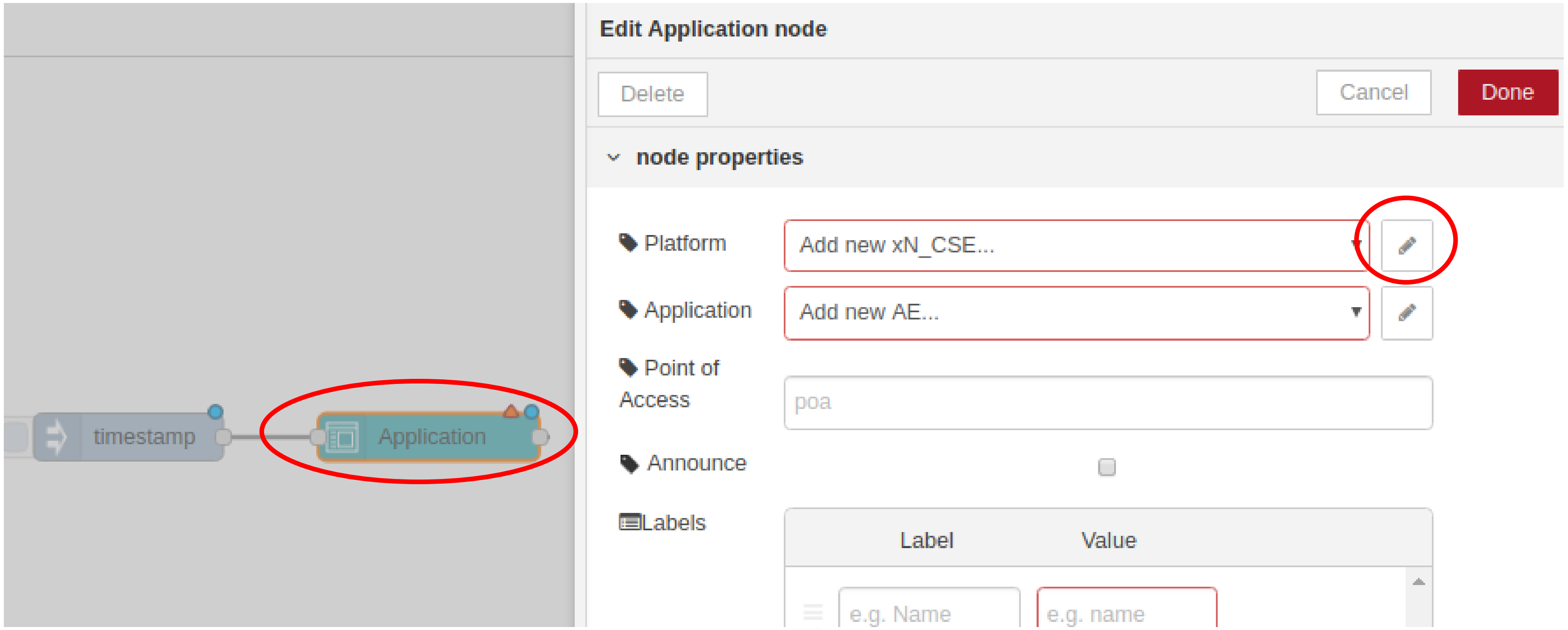
Creating a MN-AE with Node-RED Part I

4. Connect the vertices of each object (node) using a drag and drop movement. Connect all vertices as shown in the picture below.



Creating a MN-AE with Node-RED Part I

5. Double click on the “Application” object. A new window is shown. Click on the pen icon (edit xN_CSE attribute contents).

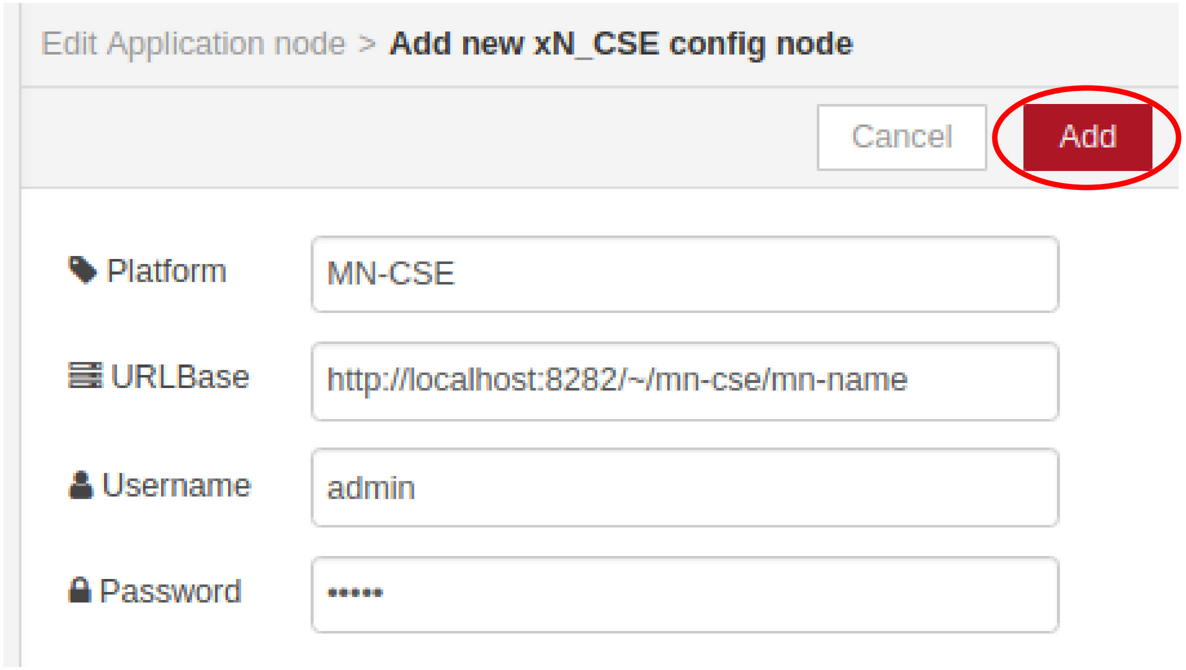


Platform	Add new xN_CSE...
Application	Add new AE...
Point of Access	poa
Announce	☐
Labels	

Label	Value
e.g. Name	e.g. name

Creating a MN-AE with Node-RED Part I

6. Complete the form "Add new xN_CSE node" according to the information shown in the picture below. Password is admin.

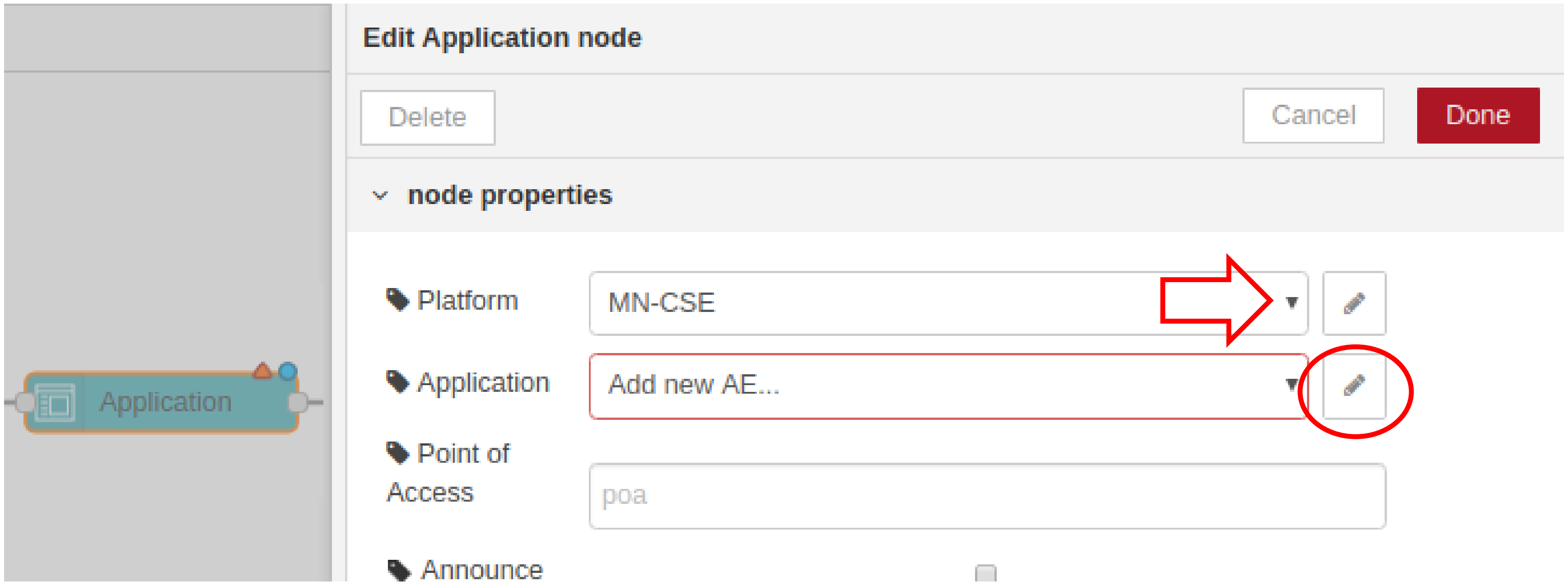


Edit Application node > **Add new xN_CSE config node**

Platform	MN-CSE
URLBase	http://localhost:8282/~/mn-cse/mn-name
Username	admin
Password	•••••

Creating a MN-AE with Node-RED Part I

7. Choose “MN-CSE” from the xN_CSE list (You just created this item in previous step).
8. Click on the pen icon for “Adding new AE…”.



Edit Application node

Delete | Cancel | Done

node properties

Platform: MN-CSE

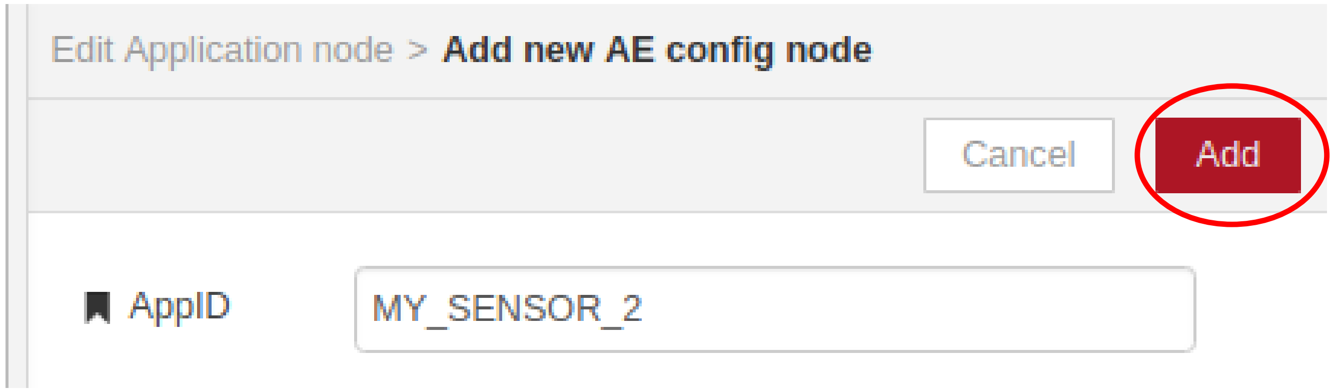
Application: Add new AE...

Point of Access: poa

Announce

Creating a MN-AE with Node-RED Part I

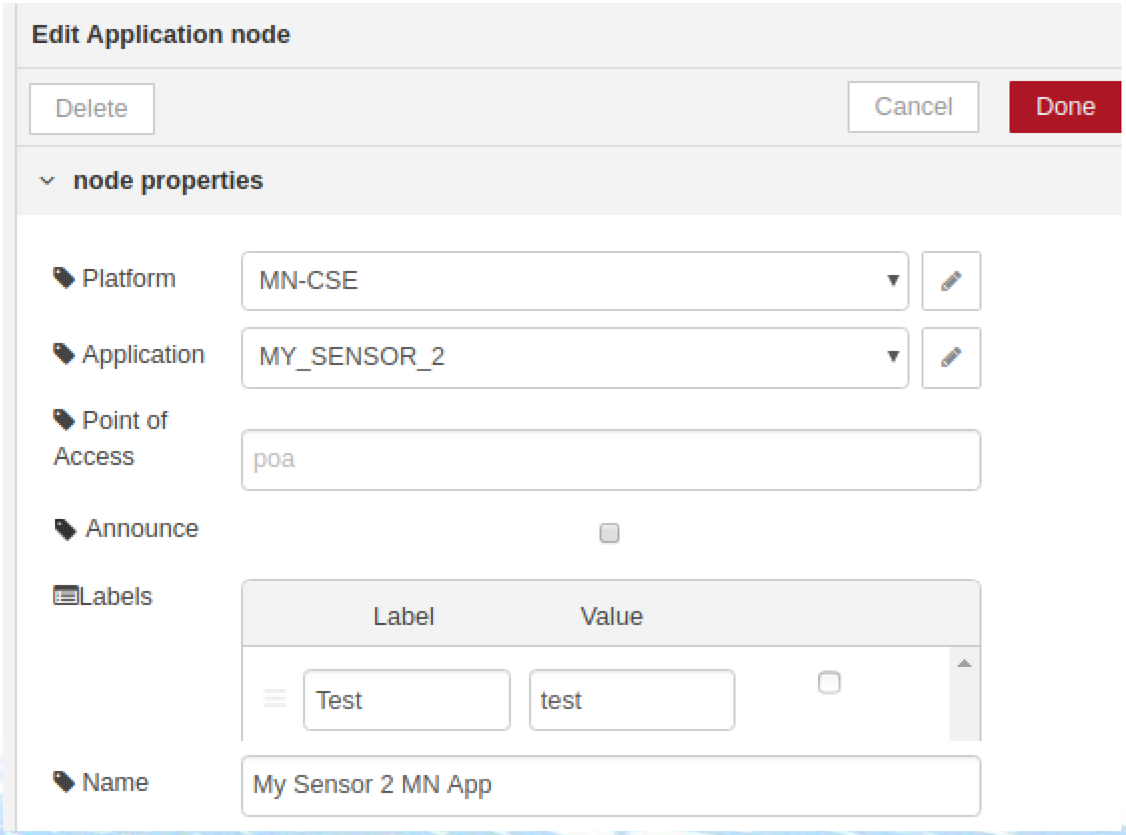
9. Complete the form “Add new NM config node” with the information shown in the picture below. Click Add.


Edit Application node > **Add new AE config node**

AppID: MY_SENSOR_2

Creating a MN-AE with Node-RED Part I

10. Choose “MY_SENSOR_2” from the Application list (You just created this item in previous step).
11. Complete the form “Edit Application node” with the information shown in the picture below.



Edit Application node

node properties

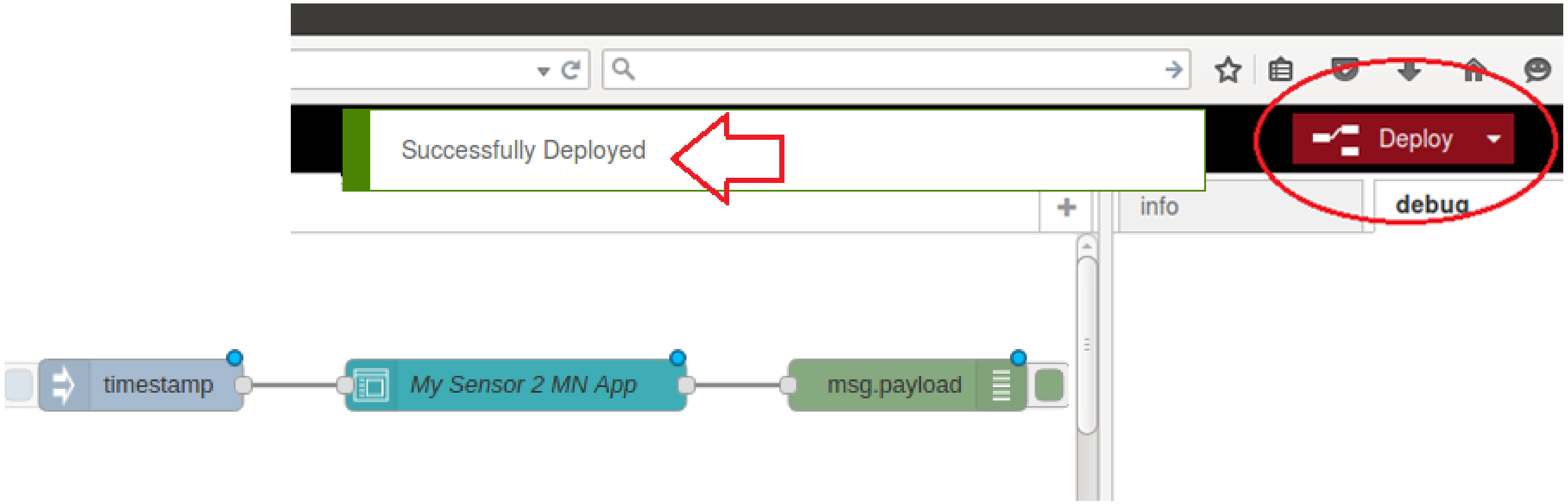
Platform	MN-CSE
Application	MY_SENSOR_2
Point of Access	poa
Announce	☐
Name	My Sensor 2 MN App

Labels

Label	Value
Test	test

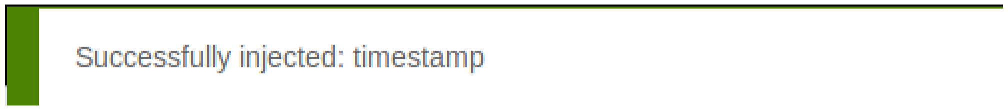
Creating a MN-AE with Node-RED Part I

12. Click on "Deploy" to compile and build you application.

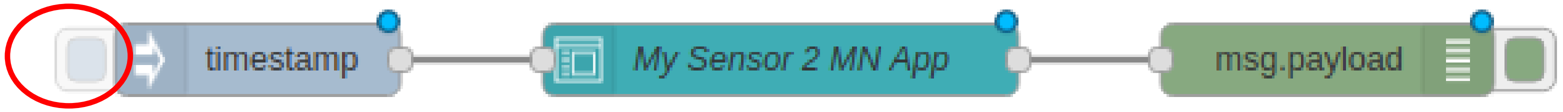


Creating a MN-AE with Node-RED Part I

13. Click on the button shown below in order to **execute** our application.

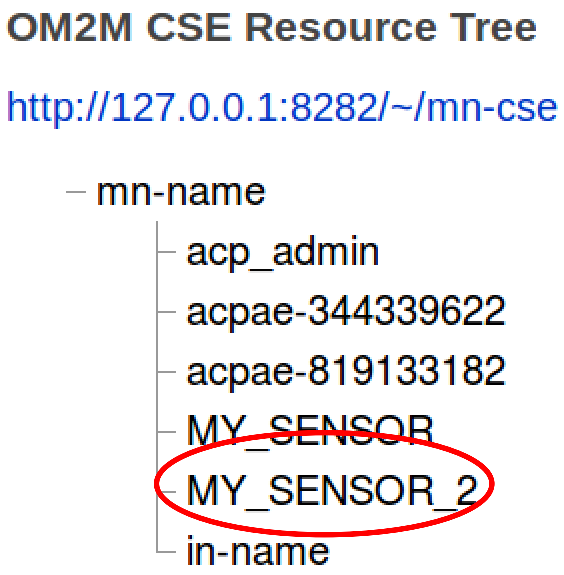

Successfully injected: timestamp

Click Here!



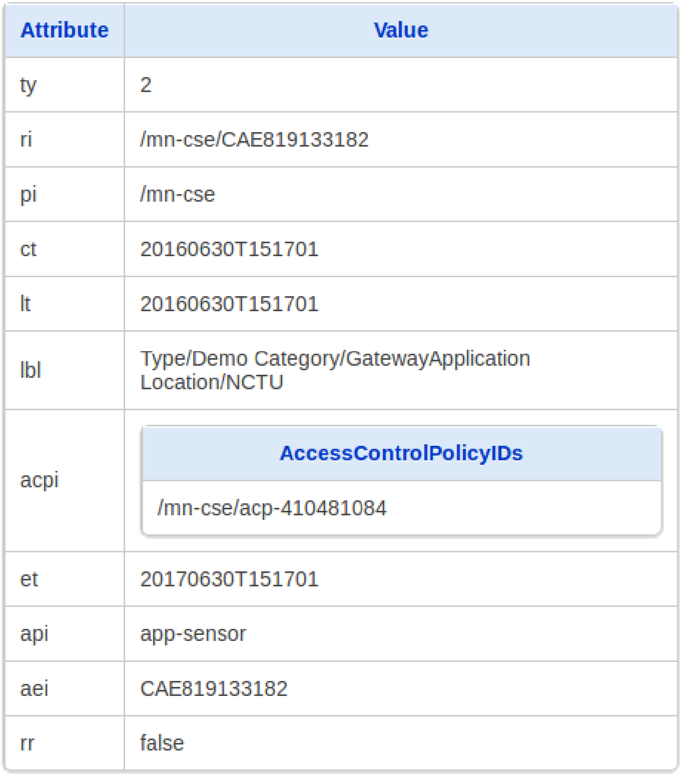
Creating a MN-AE with Node-RED Part I

14. Open OM2M Resource Tree Navigation Tool for MN-CSE and find our newly created application.


OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse

```
– mn-name
    ├ acp_admin
    ├ acpae-344339622
    ├ acpae-819133182
    ├ MY_SENSOR
    ├ MY_SENSOR_2
    └ in-name
```



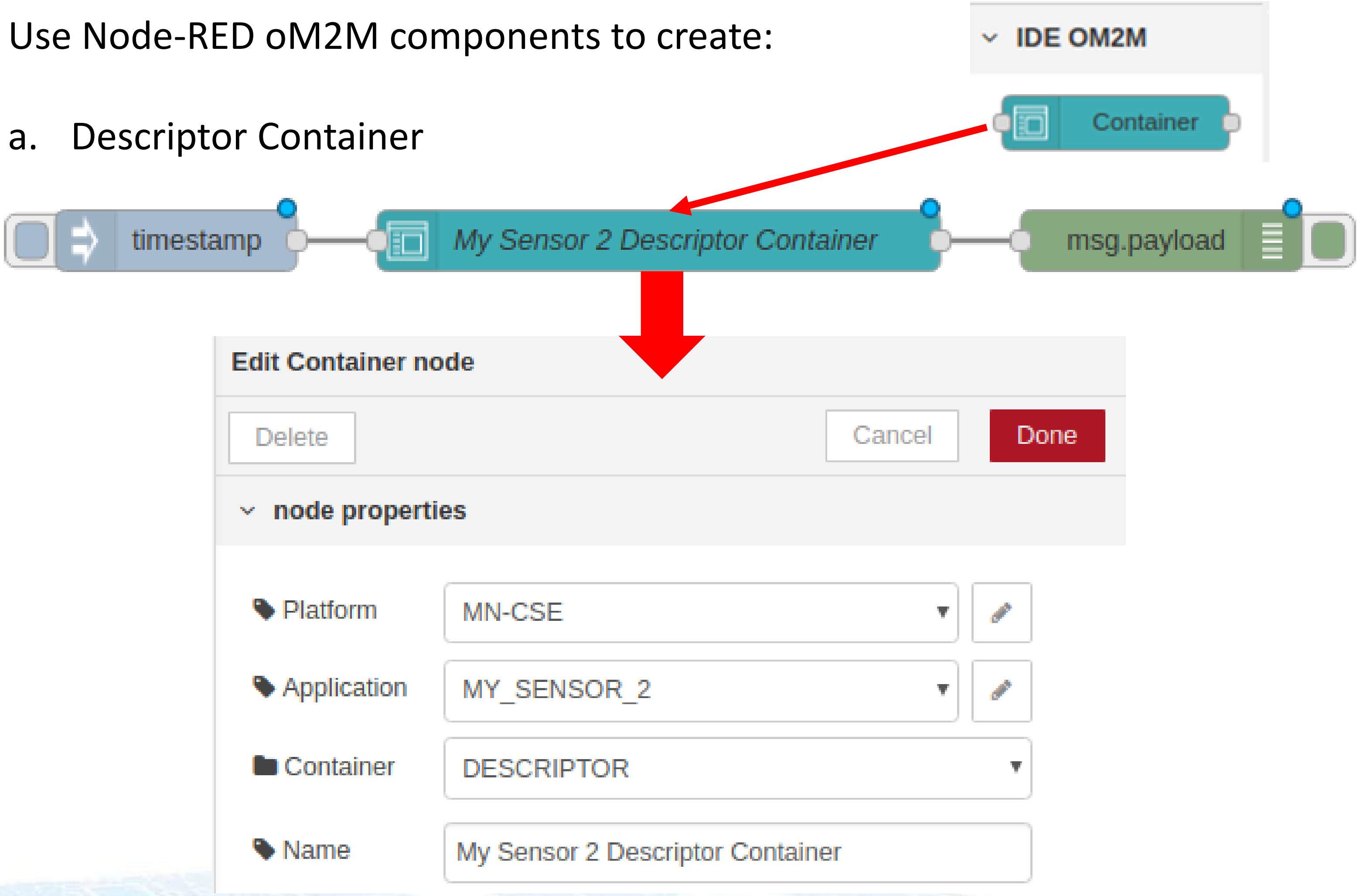
Attribute	Value
ty	2
ri	/mn-cse/CAE819133182
pi	/mn-cse
ct	20160630T151701
lt	20160630T151701
lbl	Type/Demo Category/GatewayApplication Location/NCTU
acpi	AccessControlPolicyIDs: /mn-cse/acp-410481084
et	20170630T151701
api	app-sensor
aei	CAE819133182
rr	false

CHECKPOINT 3!

Creating a MN-AE with Node-RED Part II

Use Node-RED oM2M components to create:

a. Descriptor Container



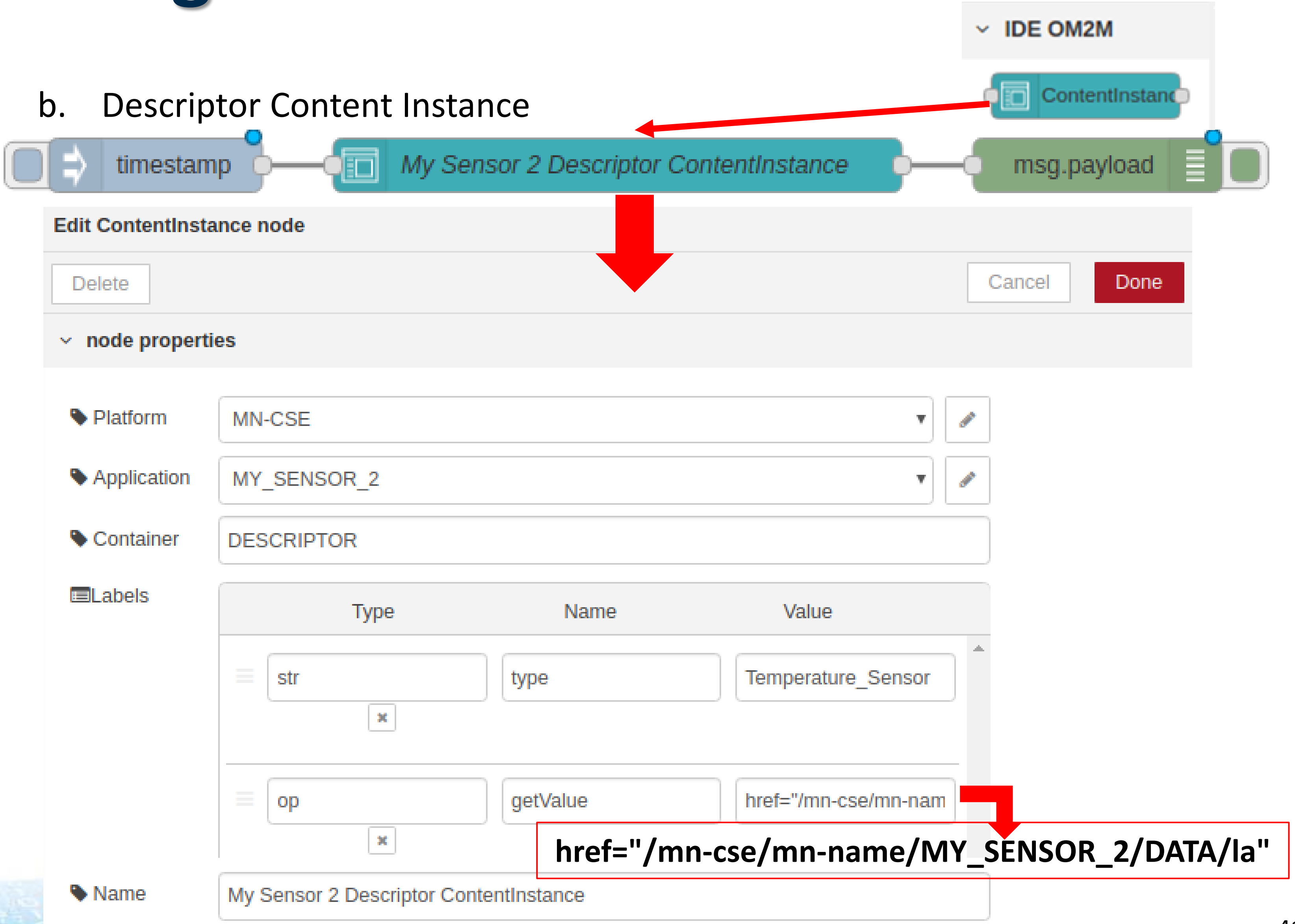
Edit Container node

node properties

Platform	MN-CSE
Application	MY_SENSOR_2
Container	DESCRIPTOR
Name	My Sensor 2 Descriptor Container

Creating a MN-AE with Node-RED Part II

b. Descriptor Content Instance



Edit ContentInstance node

Platform	MN-CSE
Application	MY_SENSOR_2
Container	DESCRIPTOR

Labels

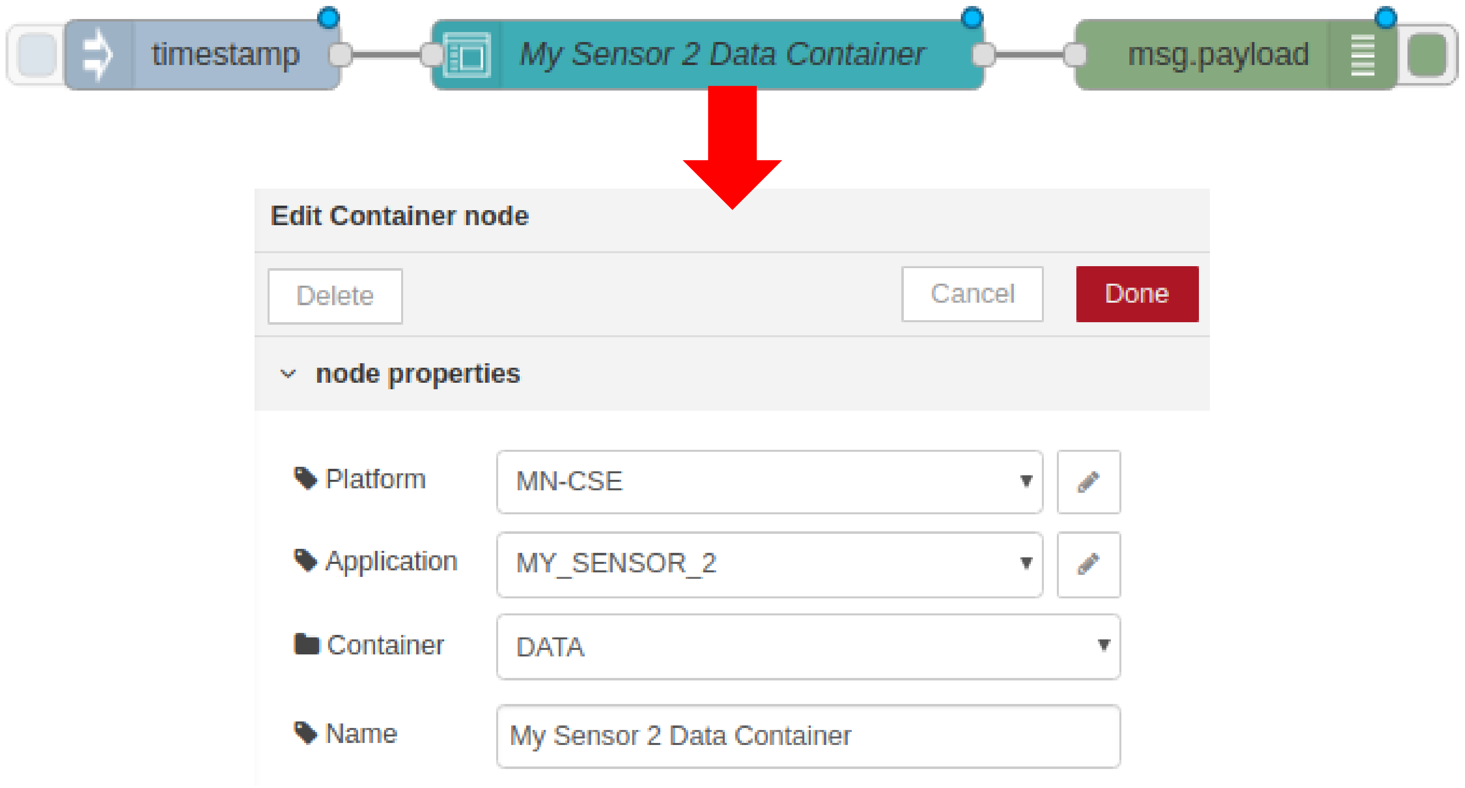
Type	Name	Value
str	type	Temperature_Sensor
op	getValue	href="/mn-cse/mn-nam

href="/mn-cse/mn-name/MY_SENSOR_2/DATA/la"

Name	My Sensor 2 Descriptor ContentInstance

Creating a MN-AE with Node-RED Part II

c. Data Container



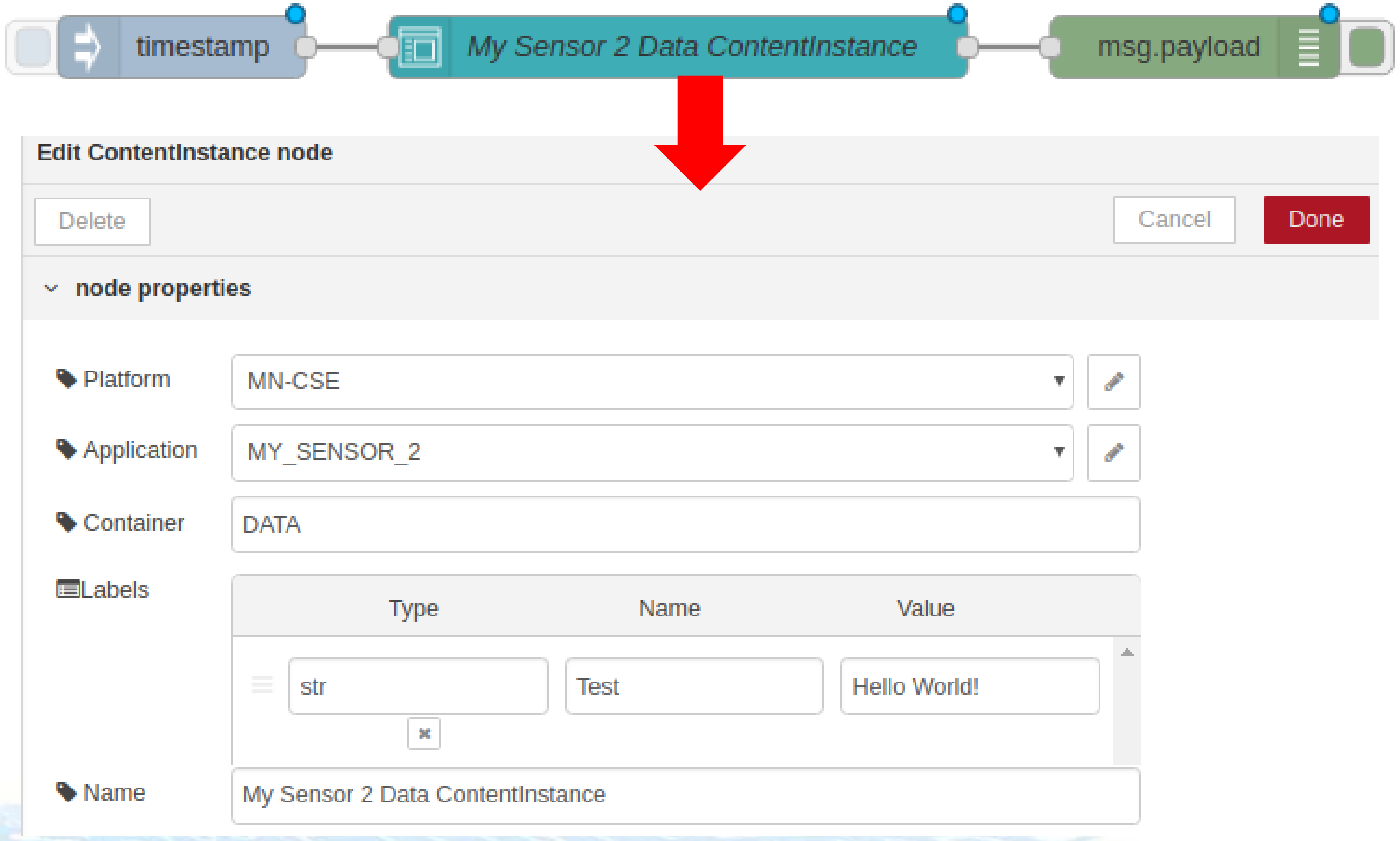
Edit Container node

node properties

Platform	MN-CSE
Application	MY_SENSOR_2
Container	DATA
Name	My Sensor 2 Data Container

Creating a MN-AE with Node-RED Part II

d. Data Content Instance



Edit ContentInstance node

Delete | Cancel | Done

node properties

Platform	MN-CSE
Application	MY_SENSOR_2
Container	DATA

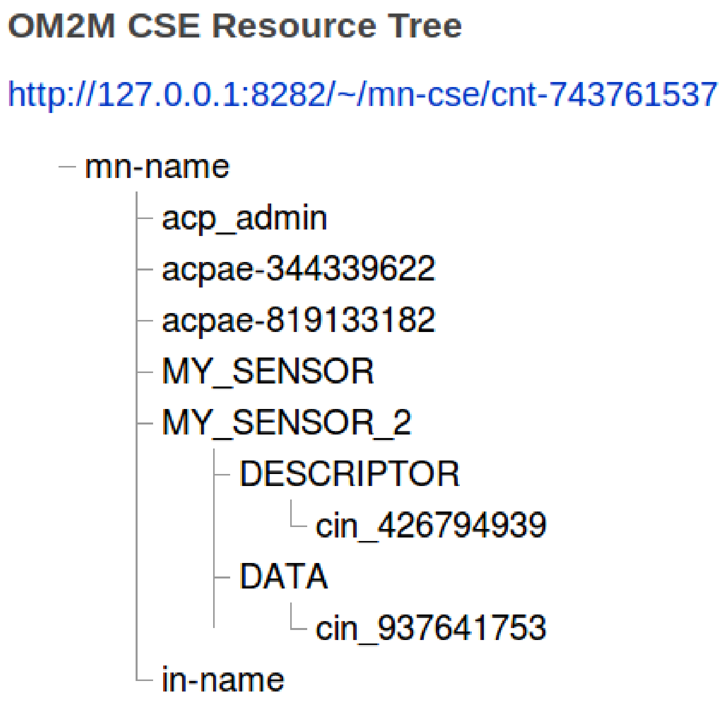
Labels

Type	Name	Value
str	Test	Hello World!

Name	My Sensor 2 Data ContentInstance

Creating a MN-AE with Node-RED Part II

Deploy the nodes again, trigger each of the flows, and use the OM2M Resource Tree Tool, verify that your data has been successfully saved into the MN-CSE.



OM2M CSE Resource Tree

http://127.0.0.1:8282/~/mn-cse/cnt-743761537

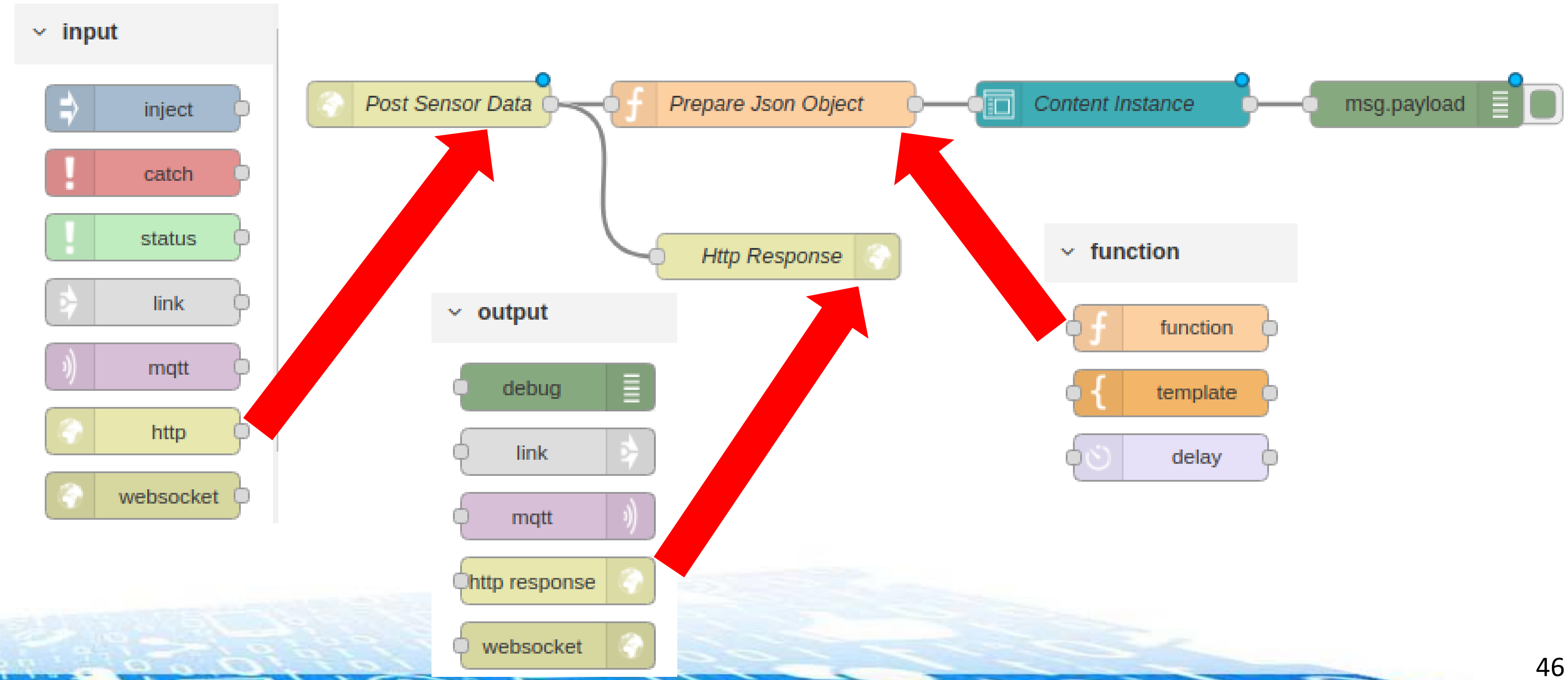
```
— mn-name
    ├ acp_admin
    ├ acpae-344339622
    ├ acpae-819133182
    ├ MY_SENSOR
    ├ MY_SENSOR_2
    │   ├ DESCRIPTOR
    │   │   └ cin_426794939
    │   ├ DATA
    │   │   └ cin_937641753
    └ in-name
```

CHECKPOINT 4!

Extending the MN-AE with HTTP Server Capabilities

1. Draw the components according to the picture.

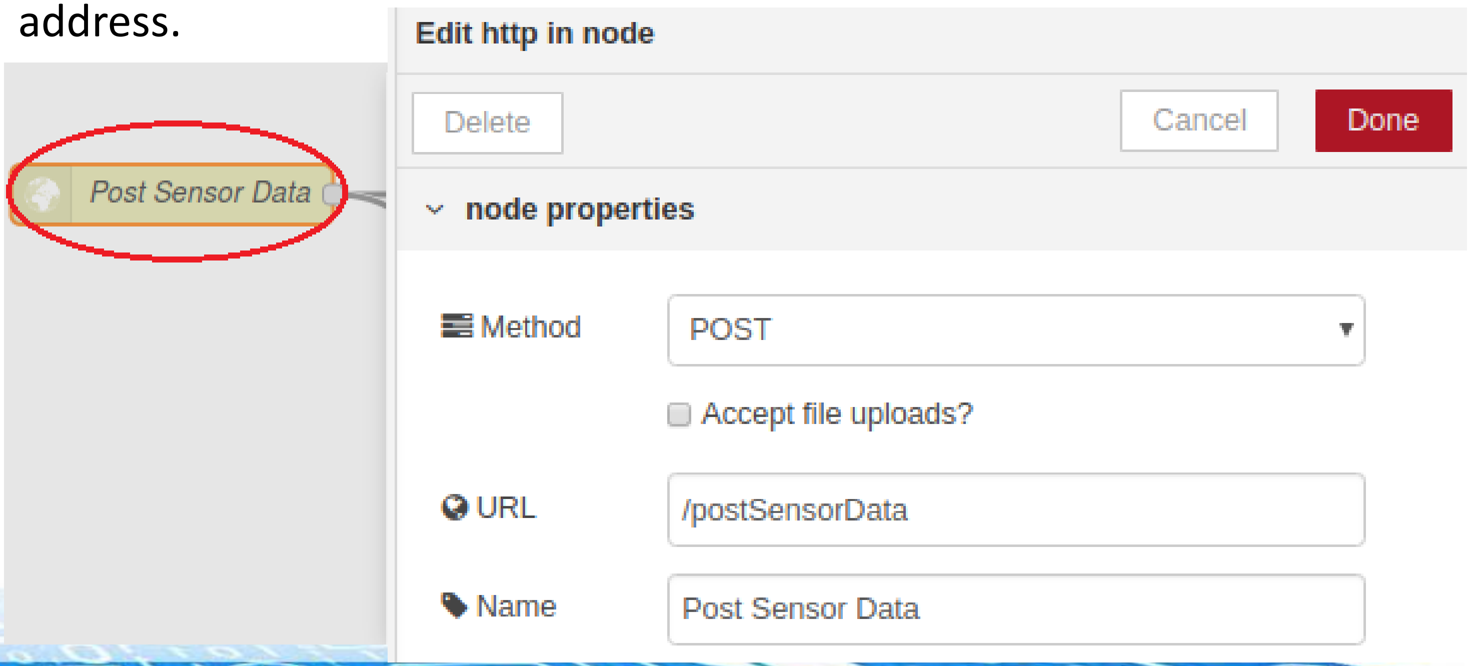
- **“http”** object from “input” box → Creates an HTTP server listening for one of the following HTTP methods: GET, POST, PUT, or DELETE.
- **“http response”** from “output” box → Returns the response to the client.



Extending the MN-AE with HTTP Server Capabilities

2. Double Click on the “Post Sensor Data” object and fill out the form according to the picture.

- With this configuration, we are creating a server that listens for HTTP POST requests in the following address: http://localhost:1880/postSensorData.
- Remember that http://localhost:1880 is the address of Node-RED. All other web services that we create are opened under the Node-RED address.



Method: POST

Accept file uploads? ☐

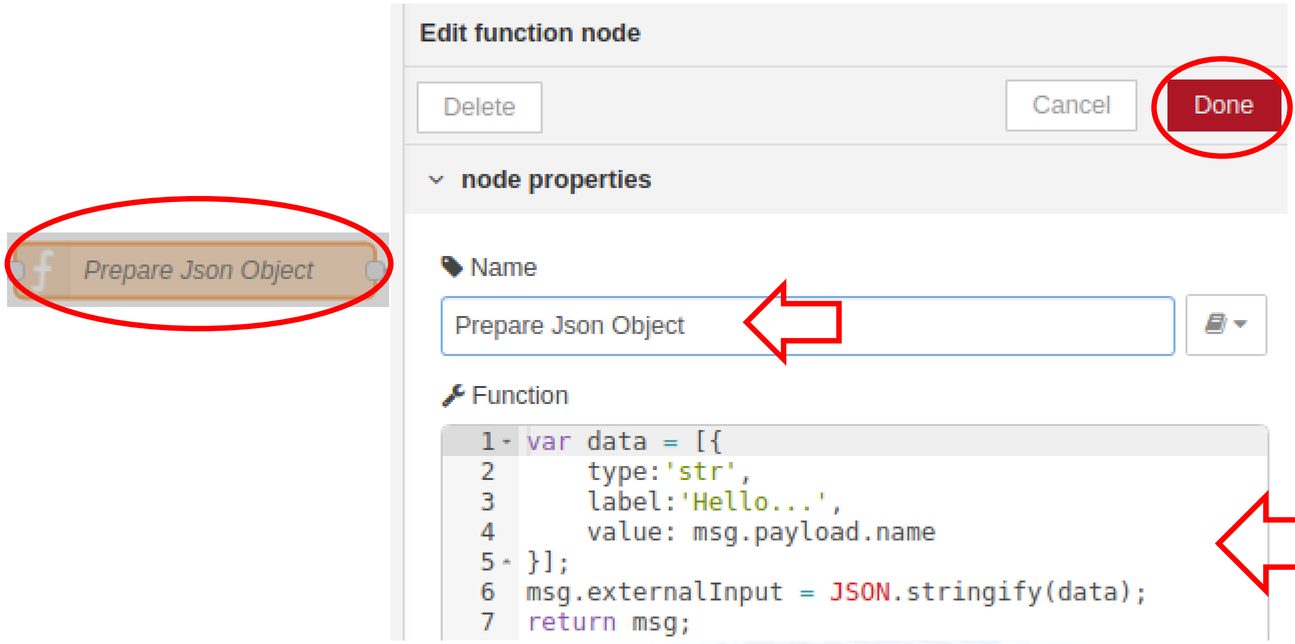
URL: /postSensorData

Name: Post Sensor Data

Extending the MN-AE with HTTP Server Capabilities

3. Double Click on the “Input Data” object and fill out the form according to the picture.

- We first arrange a Json object with the appropriate format for sending it to our platform. And then,
- We read the value of the variable “name”. This server expects a variable called “name” in the POST request.



Name: Prepare Json Object

Function

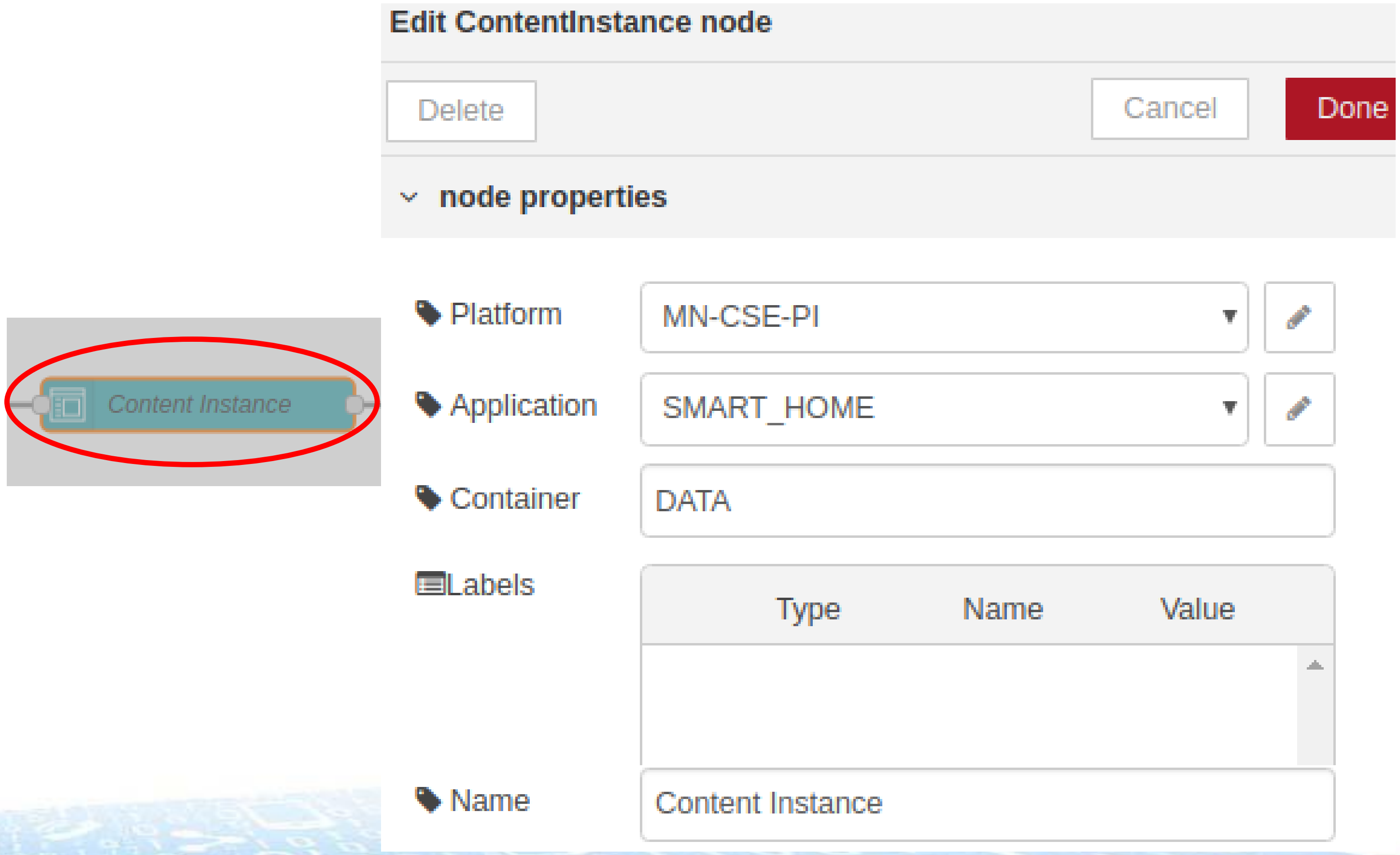
```
var data = [{
    type:'str',
    label:'Hello...',
    value: msg.payload.name
}];
msg.externalInput = JSON.stringify(data);
return msg;
```

CODE (Copy and Paste)

```
var data = [{
    type:'str',
    label:'Hello...',
    value: msg.payload.name
}];
msg.externalInput =
JSON.stringify(data);
return msg;
```

Extending the MN-AE with HTTP Server Capabilities

4. Double Click on the "Content Instance" object and fill out the form according to the picture. Notice that "Labels" is empty. You must delete any labels.



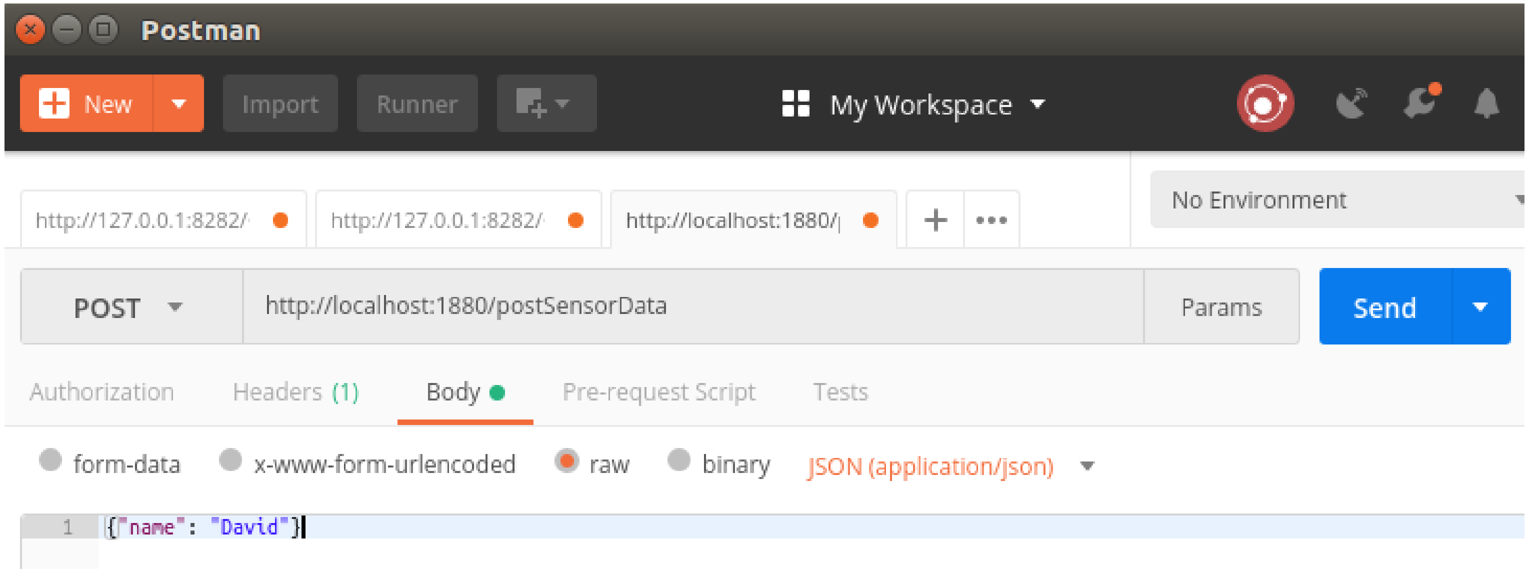
Edit ContentInstance node

Platform	MN-CSE-PI
Application	SMART_HOME
Container	DATA
Labels	Type Name Value
Name	Content Instance

Extending the MN-AE with HTTP Server Capabilities

[TASKS]

a. In order to test our server, please send an HTTP POST (using Postman) to http://localhost:1880/postSensorData. You must include a JSON object similar to this example: {"name": "TYPE YOUR NAME HERE"} in the body of the request. In Postman, set "raw" and "application/json".



POST http://localhost:1880/postSensorData

Body: raw, JSON (application/json)

```
{"name": "David"}
```

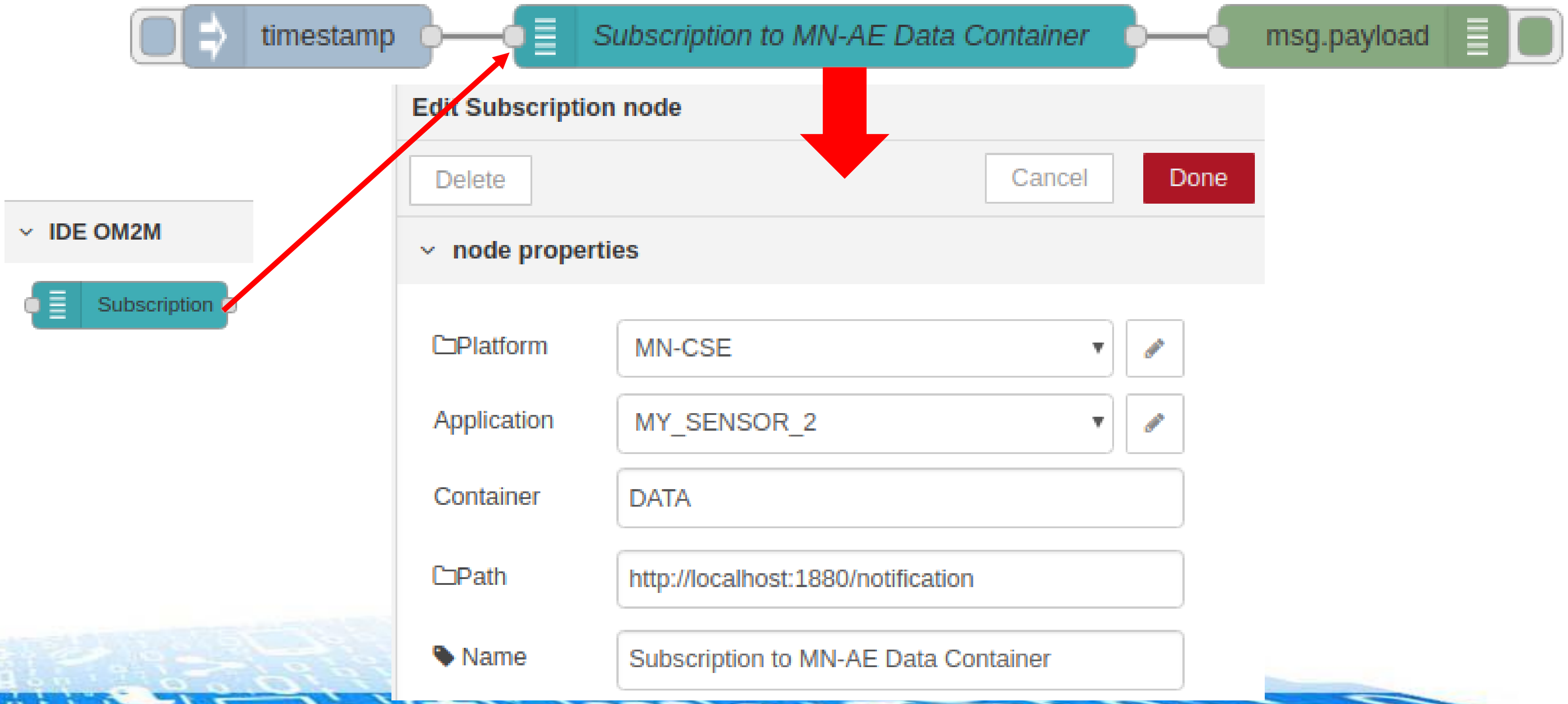
b. Use the OM2M Resource Tree Tool, verify that your data has been successfully saved into the MN-CSE.

CHECKPOINT 5!

Creating a subscription resource to a container in MN-AE using Node-RED

1. Create a subscription into the MN-AE for receiving notification when new data is saved into MN-AE.

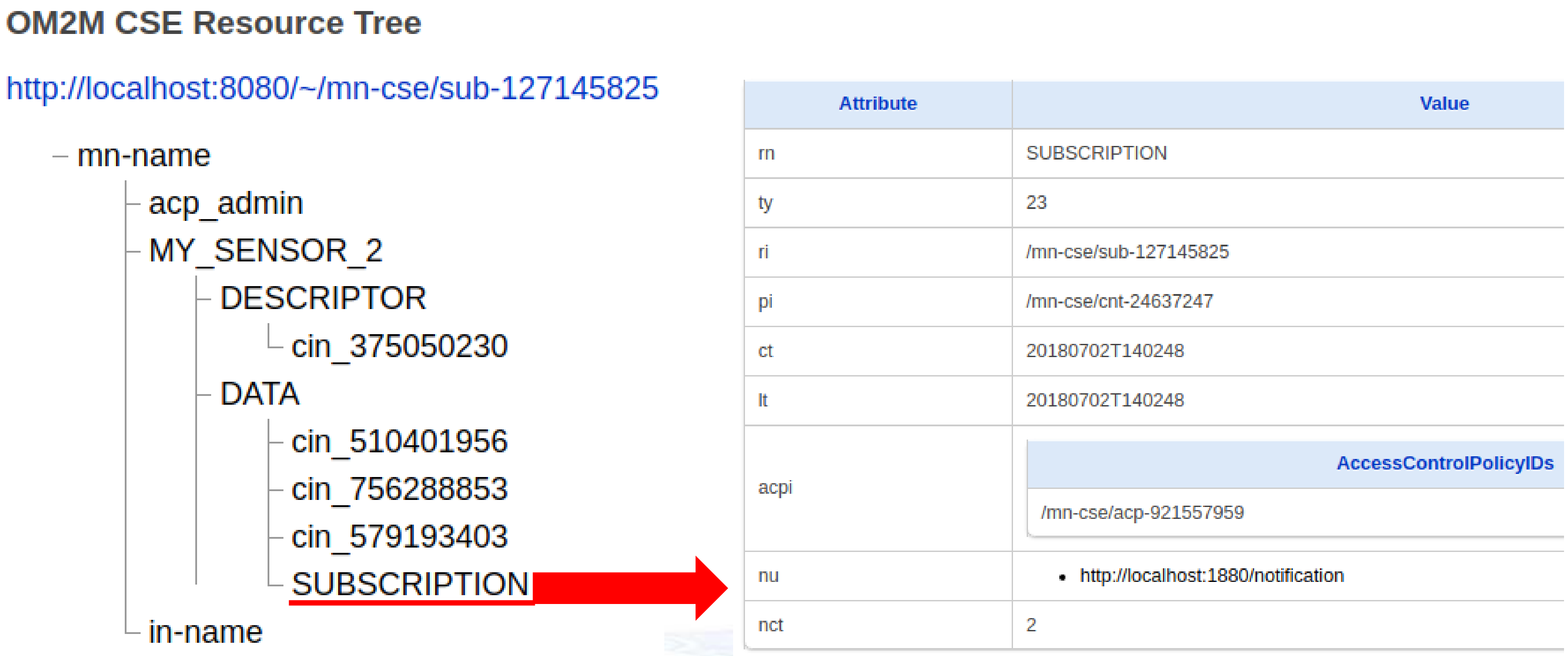
- Use a "Subscription" object from IDE OM2M to create a new subscription.
- Complete the form according to the picture below.



Platform	MN-CSE
Application	MY_SENSOR_2
Container	DATA
Path	http://localhost:1880/notification
Name	Subscription to MN-AE Data Container

Creating a subscription resource to a container in MN-AE using Node-RED

2. Deploy the project, trigger the last flow, and verify that the subscription to MY_SENSOR application has been created successfully into the MN-CSE resource tree.



OM2M CSE Resource Tree

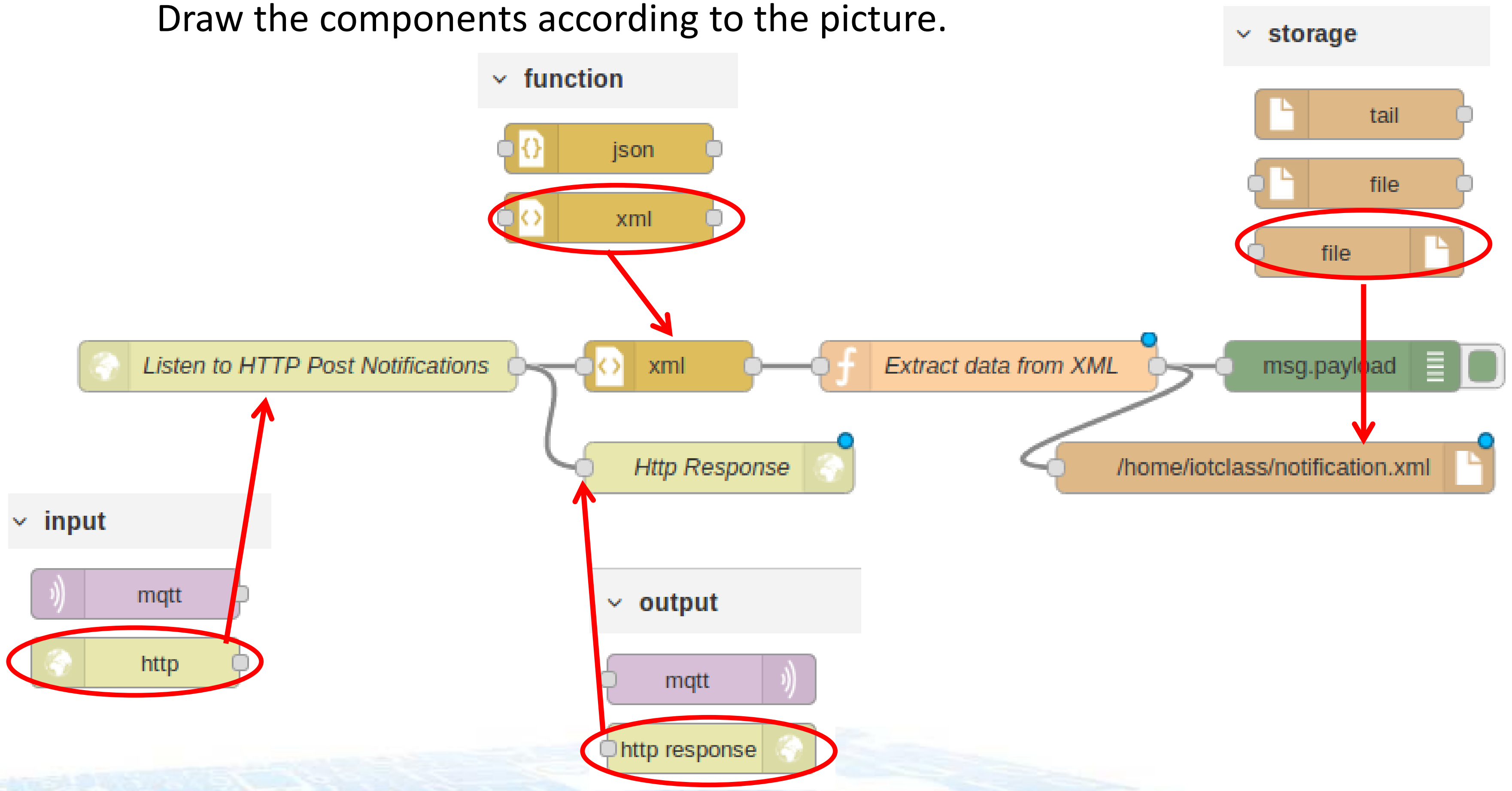
http://localhost:8080/~/mn-cse/sub-127145825

```
– mn-name
    ├ acp_admin
    ├ MY_SENSOR_2
    │   ├ DESCRIPTOR
    │   │   └ cin_375050230
    │   ├ DATA
    │   │   ├ cin_510401956
    │   │   ├ cin_756288853
    │   │   ├ cin_579193403
    │   │   └ SUBSCRIPTION
    └ in-name
```

Attribute	Value
rn	SUBSCRIPTION
ty	23
ri	/mn-cse/sub-127145825
pi	/mn-cse/cnt-24637247
ct	20180702T140248
lt	20180702T140248
acpi	AccessControlPolicyIDs: /mn-cse/acp-921557959
nu	• http://localhost:1880/notification
nct	2

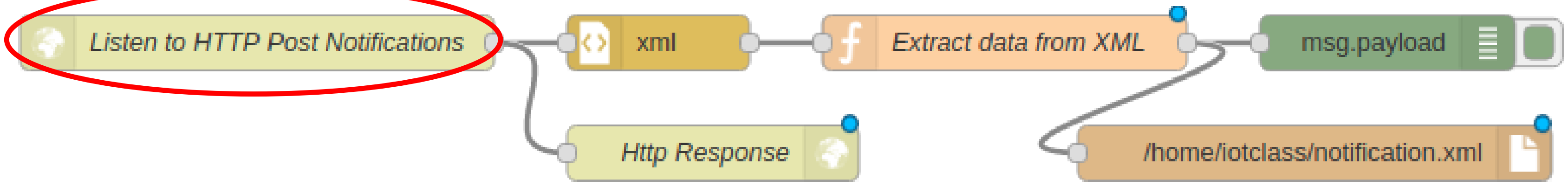
Creating a subscription resource to a container in MN-AE using Node-RED

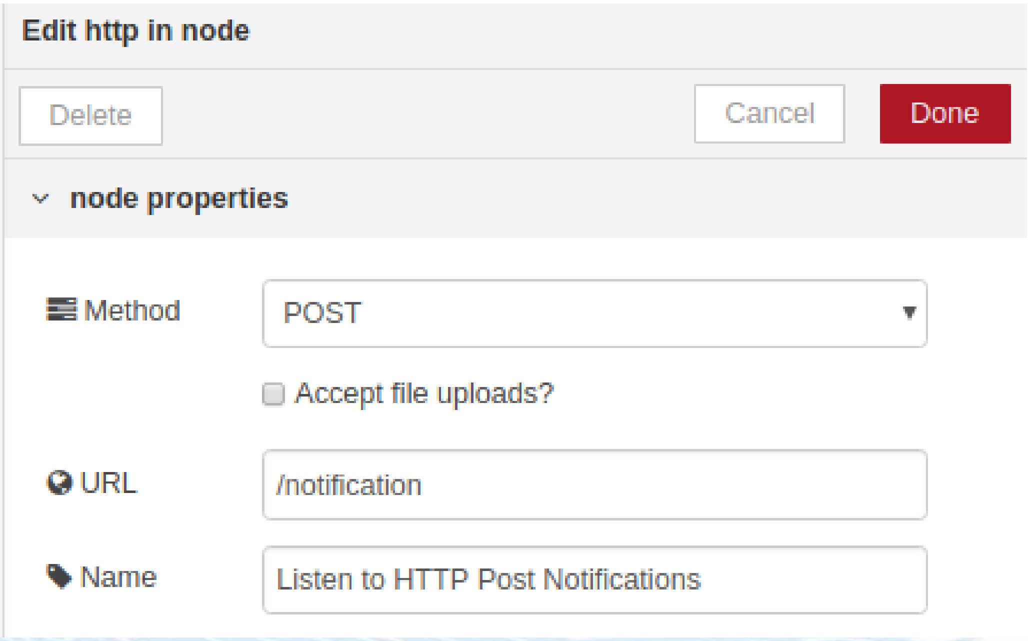
3. Create a web service listening for notifications.
Draw the components according to the picture.



Creating a subscription resource to a container in MN-AE using Node-RED

4. Double click on the "Listen to HTTP Post notifications" object and complete its form according to the picture.


Listen to HTTP Post Notifications
xml
Extract data from XML
msg.payload
Http Response
/home/iotclass/notification.xml



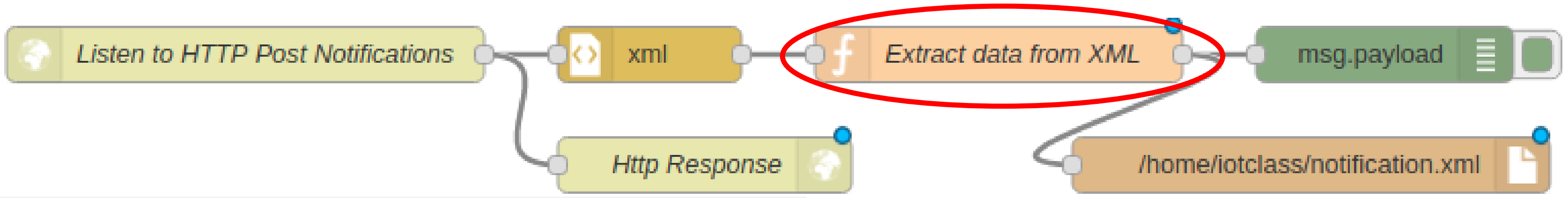
Edit http in node

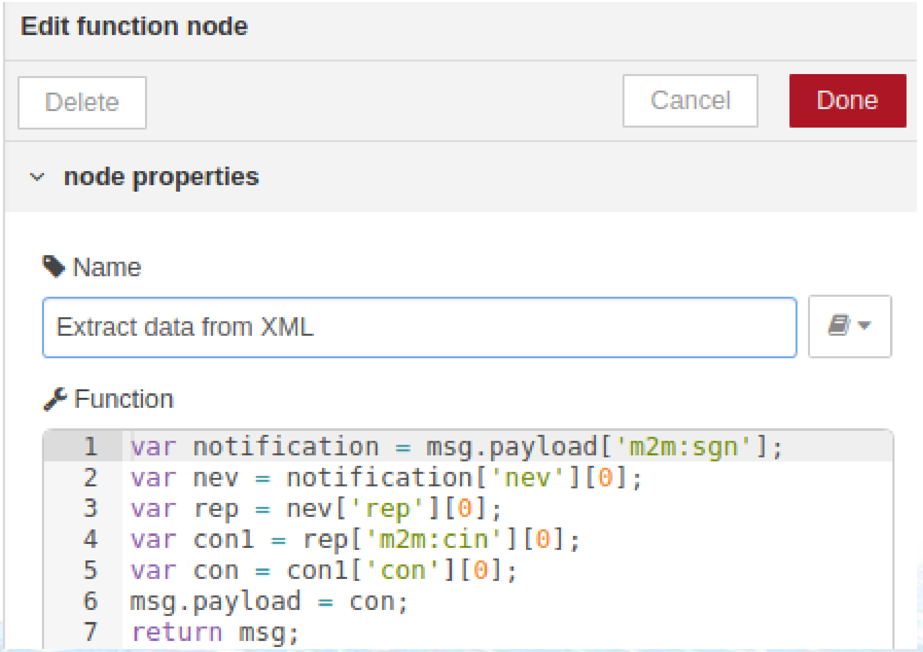
node properties

Method	POST
	☐ Accept file uploads?
URL	/notification
Name	Listen to HTTP Post Notifications

Creating a subscription resource to a container in MN-AE using Node-RED

5. Double click on the “Extract Data from XML” object and complete its form according to the picture.


Listen to HTTP Post Notifications
xml
Extract data from XML
msg.payload
Http Response
/home/iotclass/notification.xml



Name: Extract data from XML

Function:

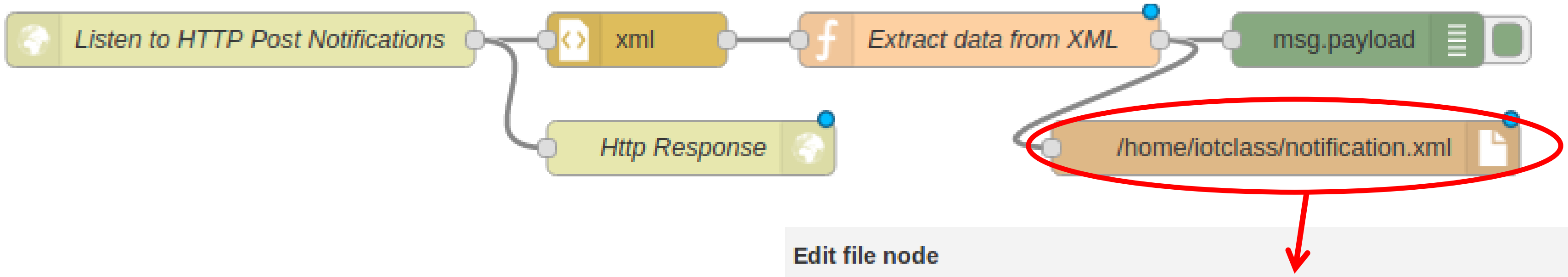
```
var notification = msg.payload['m2m:sgn'];
var nev = notification['nev'][0];
var rep = nev['rep'][0];
var con1 = rep['m2m:cin'][0];
var con = con1['con'][0];
msg.payload = con;
return msg;
```

CODE (Copy and Paste)

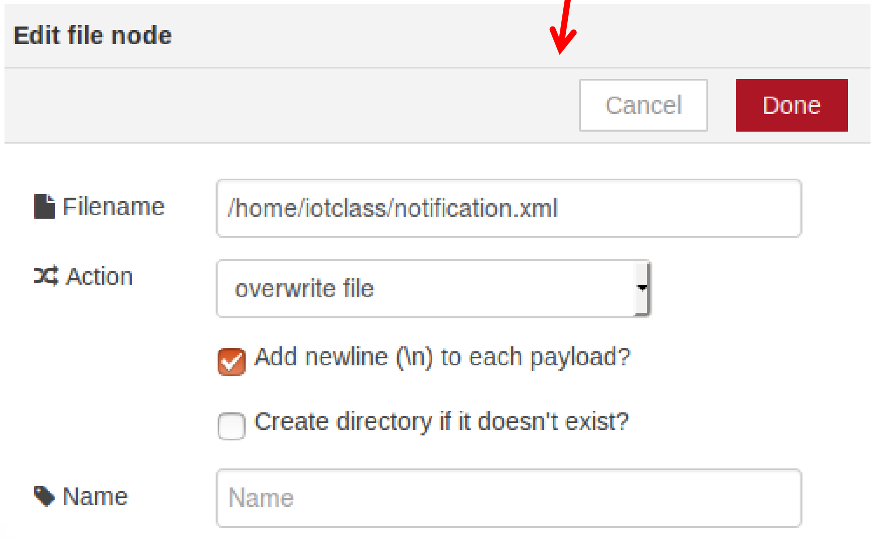
```
var notification = msg.payload['m2m:sgn'];
var nev = notification['nev'][0];
var rep = nev['rep'][0];
var con1 = rep['m2m:cin'][0];
var con = con1['con'][0];
msg.payload = con;
return msg;
```

Creating a subscription resource to a container in MN-AE using Node-RED

6. Double click on the "/home/iotclass/notification.xml" object and complete its form according to the picture.


Listen to HTTP Post Notifications
xml
Extract data from XML
msg.payload
Http Response
/home/iotclass/notification.xml

7. Use the Filename path: "/home/iotclass/notification.xml".
8. The remaining objects don't need extra configuration.



Filename: /home/iotclass/notification.xml
Action: overwrite file
[✓] Add newline (\n) to each payload?
[ ] Create directory if it doesn't exist?
Name: Name

Creating a subscription resource to a container in MN-AE using Node-RED

- At this moment we have created a web service which is listening to HTTP POST requests in the following address: http://localhost:1880/notification.

- Every time a new ContentInstance is created into the DATA container of "MY_SENSOR_2" MN-AE (new data is inserted), a notification message will be sent automatically to the address listed above.

Remember:
In order to insert new data to "MY_SENSOR_2" MN-AE make an HTTP POST (using Postman) to the address http://localhost:1880/postSensorData including a JSON object similar to {"name": "TYPE YOUR NAME HERE"}.
(You already did this on slide 50).

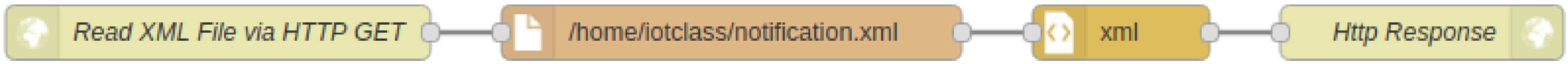
Creating a subscription resource to a container in MN-AE using Node-RED

[TASKS]

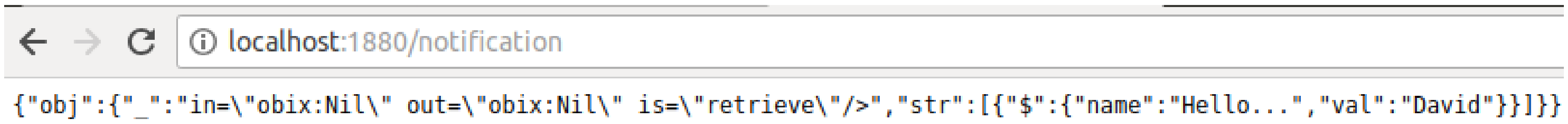
- Extend your Network Application, creating an HTTP GET web service in order to retrieve the content of the XML file which is the result of the previous exercise.

Hints:

- Use the following components of Node-RED



- Your output should be similar to the picture in the below.



```
{"obj":{"_":"in=\"obix:Nil\" out=\"obix:Nil\" is=\"retrieve\"/>","str":[{"$":{"name":"Hello...","val":"David"}}]}}
```

CHECKPOINT 6!

Appendix

OM2M
Connecting things

OM2M Overview (1)

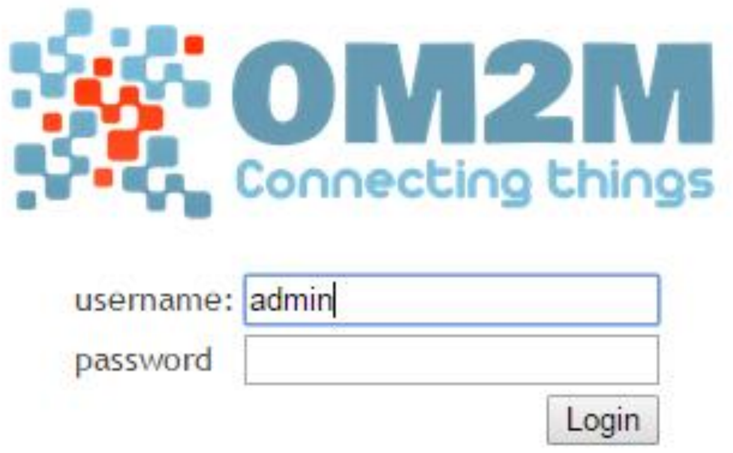
http://www.eclipse.org/om2m/

- The Eclipse OM2M project, initiated by LAAS-CNRS, is an open source implementation of oneM2M and SmartM2M standard.
- It provides a **horizontal Common Service Entity (CSE)** that can be deployed in an M2M server, a gateway, or a device.
- Each CSE provides Application Enablement, Security, Triggering, Notification, Persistency, Device Interworking, Device Management, etc.
- Exposes a **RESTful API** providing primitive procedures for machines authentication, resources discovery, applications registration, containers management, synchronous and asynchronous communications, access rights authorization, groups organization, and re-targeting.
- OM2M is a Java implementation running on top of an OSGi Equinox.

OM2M Overview (2)

OM2M Resource Tree Visualizer Tool

- Address: http://localhost:8080/webpage
- username: admin ; password: admin



username: admin
password

OM2M Overview (3)

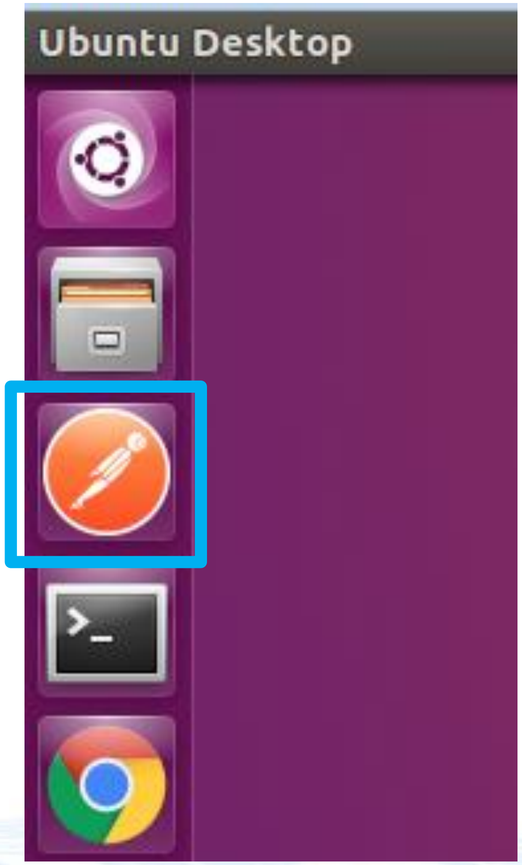
Authorization header

- OM2M supports Basic access authentication to enforce access controls to web resources.
- Client username/password must be encoded to base64 then entered as a Basic Authorization header.
- You can use http://www.base64encode.org for base64 encryption. For example: base64(admin:admin) = “YWRtaW46YWRtaW4=“.

HTTP requests with PostMan (1)

Running PostMan

Start postman on your computer:

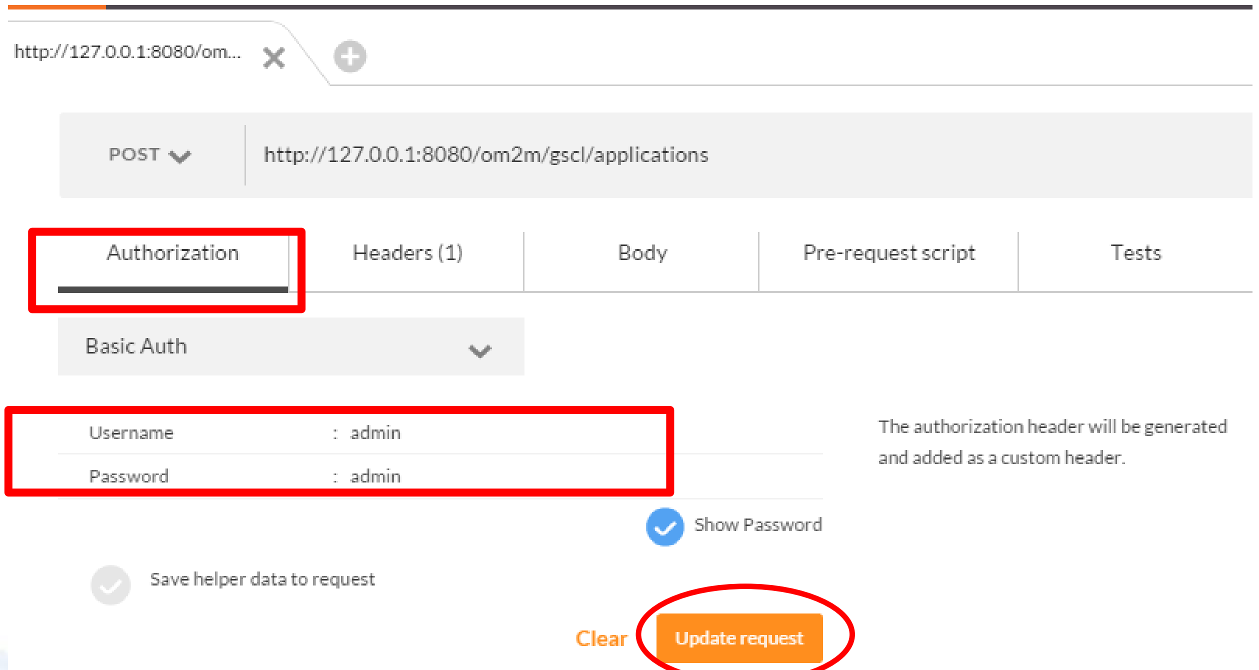


HTTP requests with PostMan (2)

Using PostMan – Setting credentials

Use postman to set authorization:

1. Select Authorization Tab, then choose “Basic Auth”.
2. Set username: admin, password: admin.
3. Click Update request.



http://127.0.0.1:8080/om2m/gscl/applications

Basic Auth

Username : admin
Password : admin

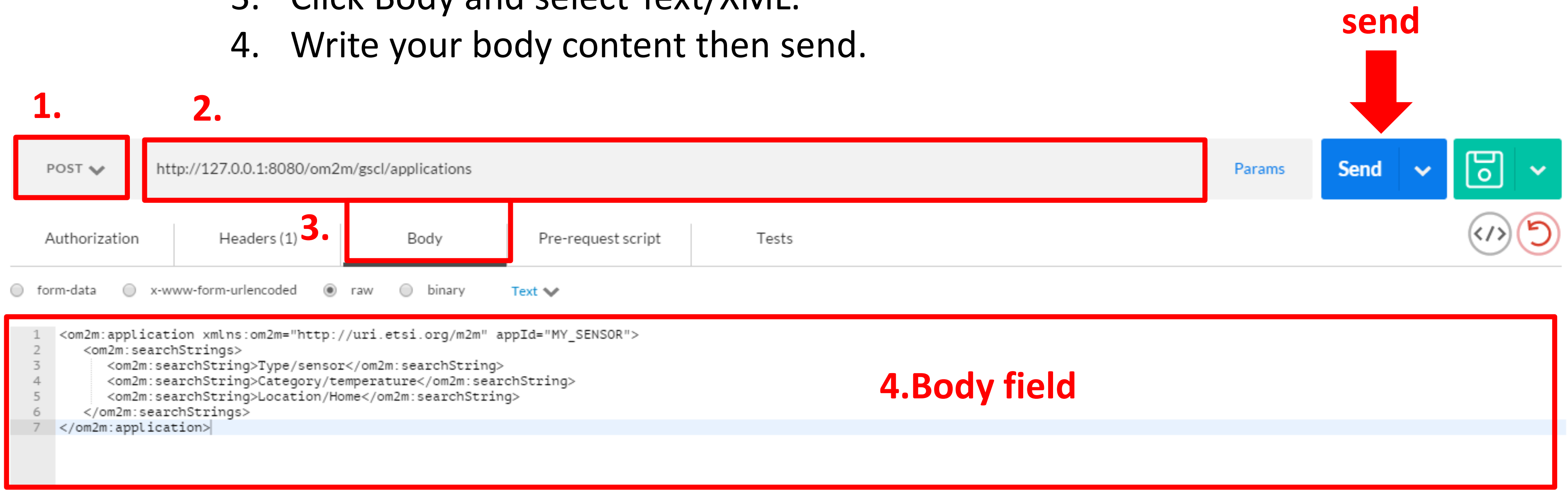
The authorization header will be generated and added as a custom header.

HTTP requests with PostMan (3)

Using PostMan – Making an HTTP POST request

Use postman tool to send an http request

1. Click method, ex post, get, put…etc.
2. Fill your OM2M URL.
3. Click Body and select Text/XML.
4. Write your body content then send.



http://127.0.0.1:8080/om2m/gscl/applications

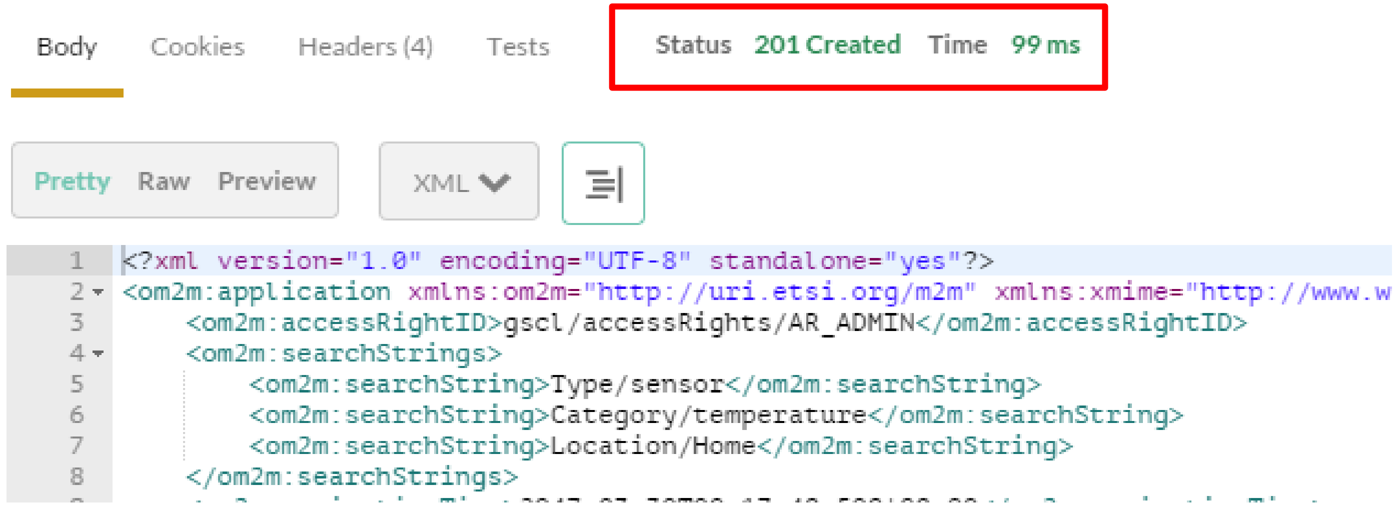
```
<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" appId="MY_SENSOR">
   <om2m:searchStrings>
      <om2m:searchString>Type/sensor</om2m:searchString>
      <om2m:searchString>Category/temperature</om2m:searchString>
      <om2m:searchString>Location/Home</om2m:searchString>
   </om2m:searchStrings>
</om2m:application>
```

HTTP requests with PostMan (4)

Using PostMan – Check request status

Use postman tool to check success or failure status:

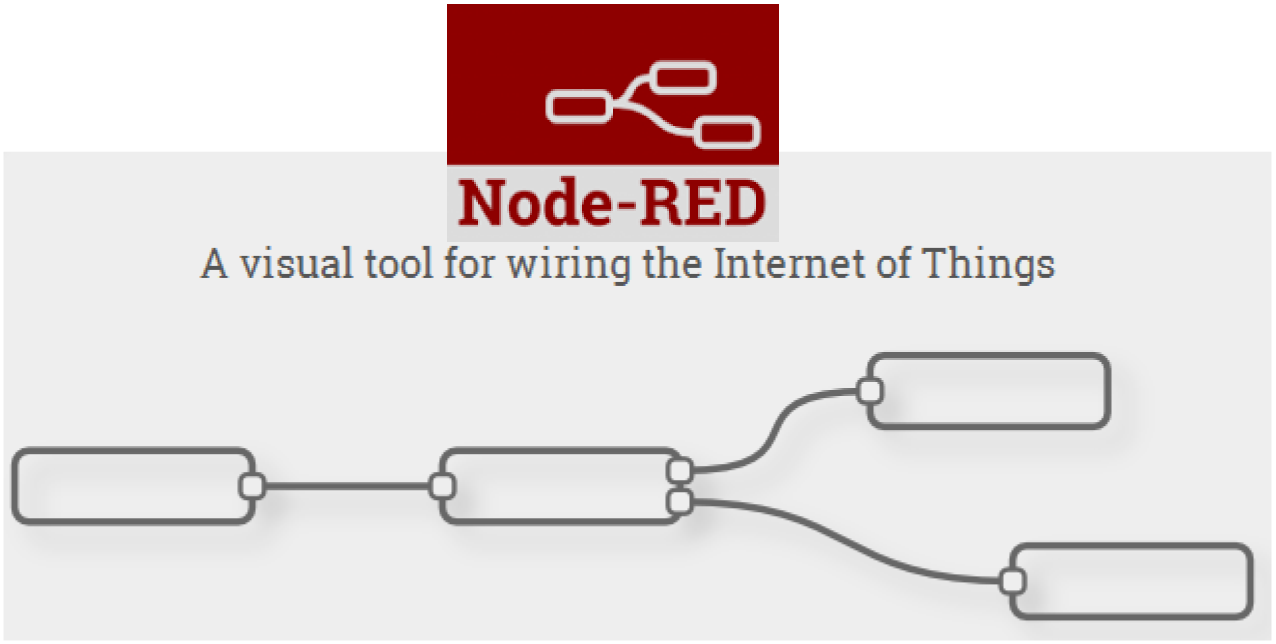
1. When send is done, it shows status and XML information on the bottom of Postman.



Status 201 Created Time 99 ms

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w
    <om2m:accessRightID>gscl/accessRights/AR_ADMIN</om2m:accessRightID>
    <om2m:searchStrings>
        <om2m:searchString>Type/sensor</om2m:searchString>
        <om2m:searchString>Category/temperature</om2m:searchString>
        <om2m:searchString>Location/Home</om2m:searchString>
    </om2m:searchStrings>
```

Node-Red



Node-Red Overview

- Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range nodes in the palette. Flows can be then deployed to the runtime in a single-click.
- JavaScript functions can be created within the editor using a rich text editor.
- A built-in library allows you to save useful functions, templates or flows for re-use.
- Based on NodeJS.
- The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.