Middleware for the IoT

TP 1 Interact with the oneM2M RESTful architecture using Eclipse OM2M

5 ISS - Ewan MACKAY / Abir BENNAZZOUZ

Launching the platform

Following the setup of the platform and checking all dependencies we launch the platform by starting an IN-CSE node :

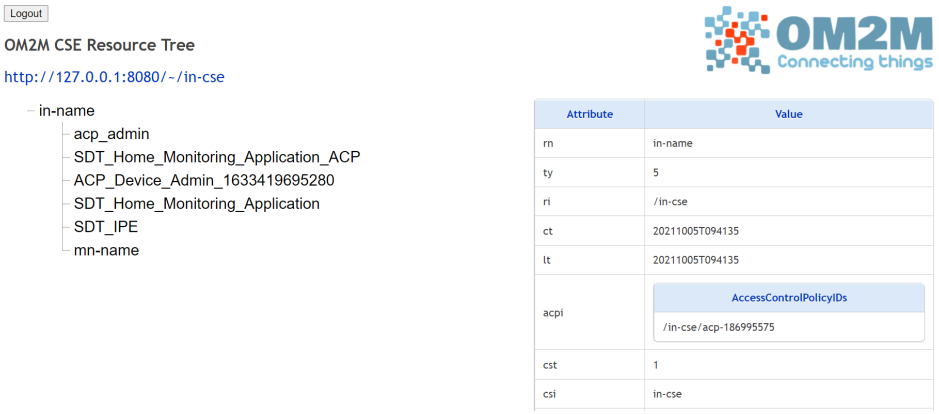
```
C:\Users\ewanm\OneDrive\Bureau\eclipse-om2m-v1-4-1\eclipse-om2m-v1-4-1\in-cse>start.bat
```

```
INFO: Starting server
Oct 13, 2021 3:54:40 PM ch.ethz.inf.vs.californium.network.CoAPEndpoint start
INFO: Starting Endpoint bound to 0.0.0.0/0.0.0.0:5683
INFO [ONEM2M.SDT] ctor
INFO [ONEM2M.SDT] Activation
INFO [ONEM2M.SDT] ctor
INFO [ONEM2M.SDT] Activation
osgi> ss
"Framework is launched."
```

We also start an MN-CSE node by using the same method. As the IN node is running, the MN node automatically authenticates to the remote IN-CSE specified in the gateway configuration file.

Using the developer interface

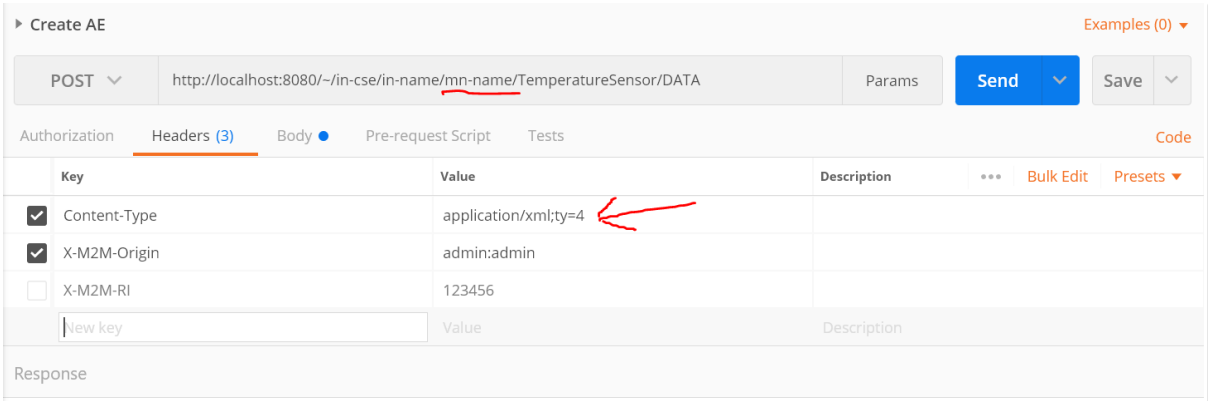
To access the developer interface, we connect to http://127.0.0.1:8080/. From here we use admin as user and password. We have not changed the default configuration and can therefore use the interface to explore the existing resources.



The MN node appears on the interface. We will now attempt to connect to the REST API to add containers and data using Postman to send HTTP requests.

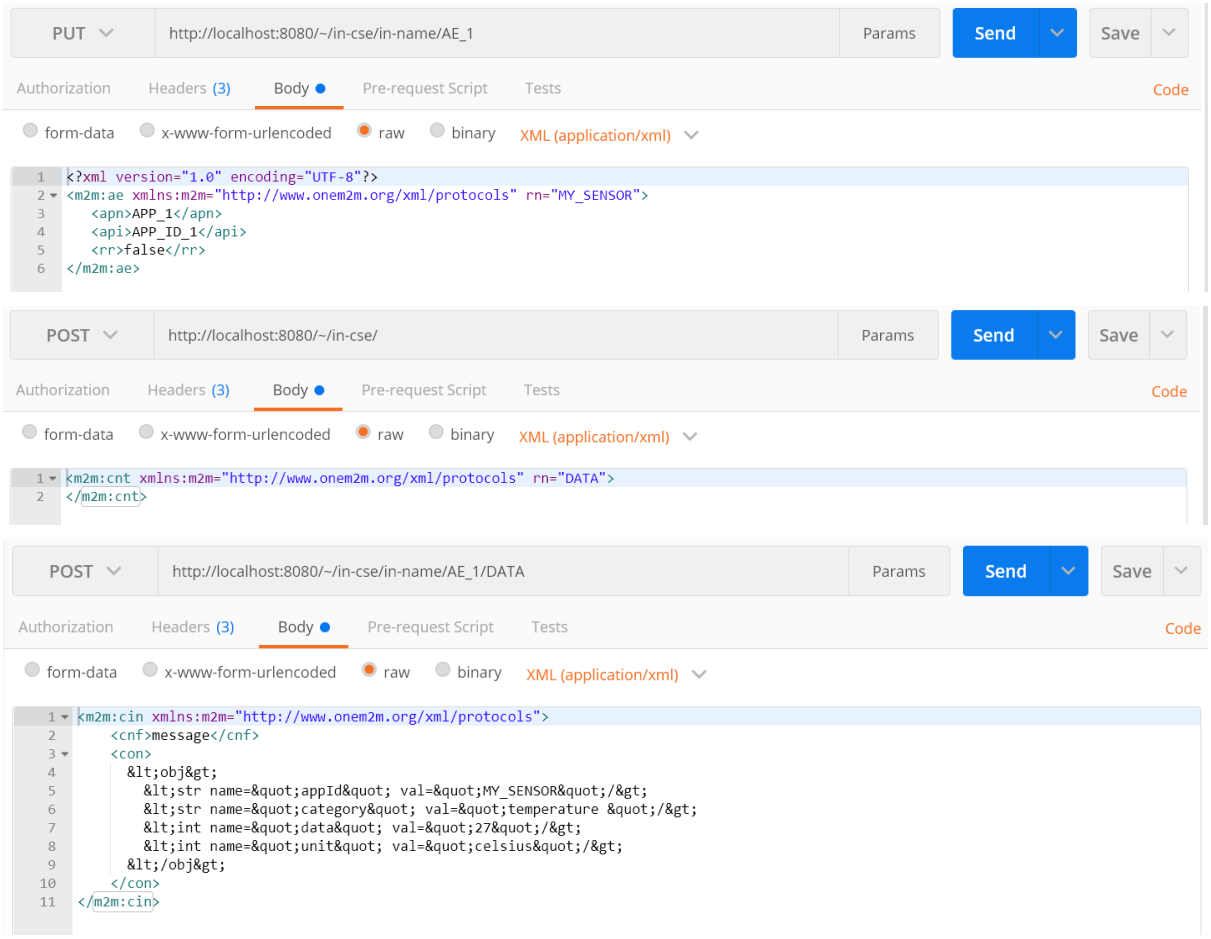
Using Postman to create HTTP requests

Postman is a software that allows us to easily create HTTP requests of the types POST, PUT, GET, DELETE methods of the REST client.These procedures correspond respectively to CREATE, UPDATE, RETRIEVE and DELETE in the resource tree. It is important to ensure that the right parameters are passed when creating requests (ty=2 for Application entities, 3 for container instances and 4 for data instances).



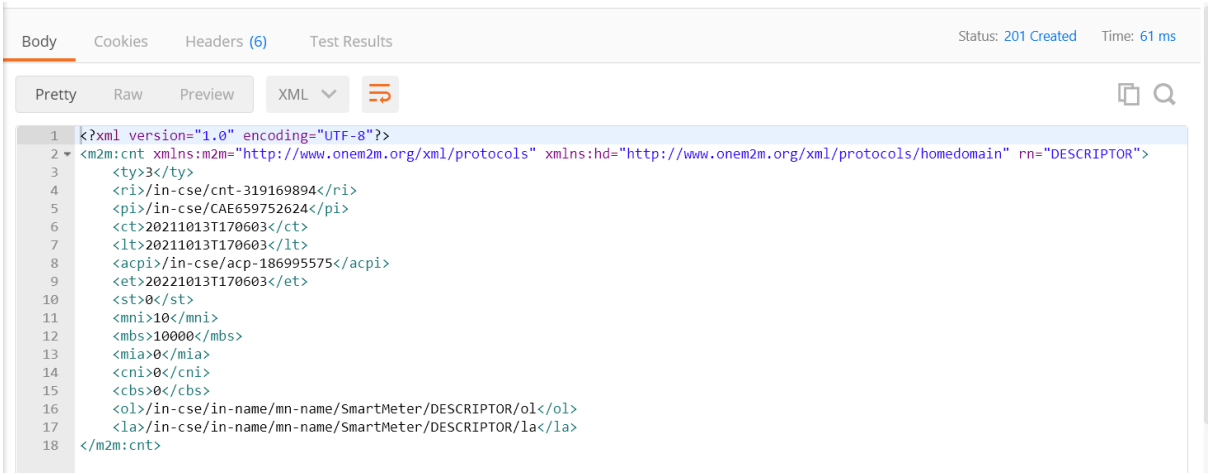
We can also modify the body of the request to change the naming of our created entities.

Here we create an application entity called MY_SENSOR in which we also create containers and data instances.



We will create three AEs, each with two containers : DATA and DESCRIPTOR.

Postman allows us to see the reply of the IN to our request (example :descriptor container creation) :



We end up with this resource tree :

- in-name
  - acp_admin
  - SDT_Home_Monitoring_Application_ACP
  - ACP_Device_Admin_1633419695280
  - SDT_Home_Monitoring_Application
  - SDT_IPE
  - mn-name
    - SmartMeter
      - DATA
      - DESCRIPTOR
    - LuminositySensor
      - DATA
      - DESCRIPTOR
    - TemperatureSensor
      - DATA
      - DESCRIPTOR

Attribute	Value
rn	TemperatureSensor
ty	2
ri	/in-cse/CAE56151324
pi	/in-cse/csr-807111461
ct	20211005T120253
lt	20211005T120253
lbl	• Category/temperature • Location/home • Type/sensor
acpi	AccessControlPolicyIDs: /in-cse/acp-186995575
et	20221005T120253
api	app-sensor
aei	CAE56151324

Access rights

With our ressource architecture in place, we will now focus on access rights.

OneM2M allows the management of access rights to resources by a user through the use of ACP resources. Each entity can be assigned one or more ACP resources which will determine the access rights of the different users :

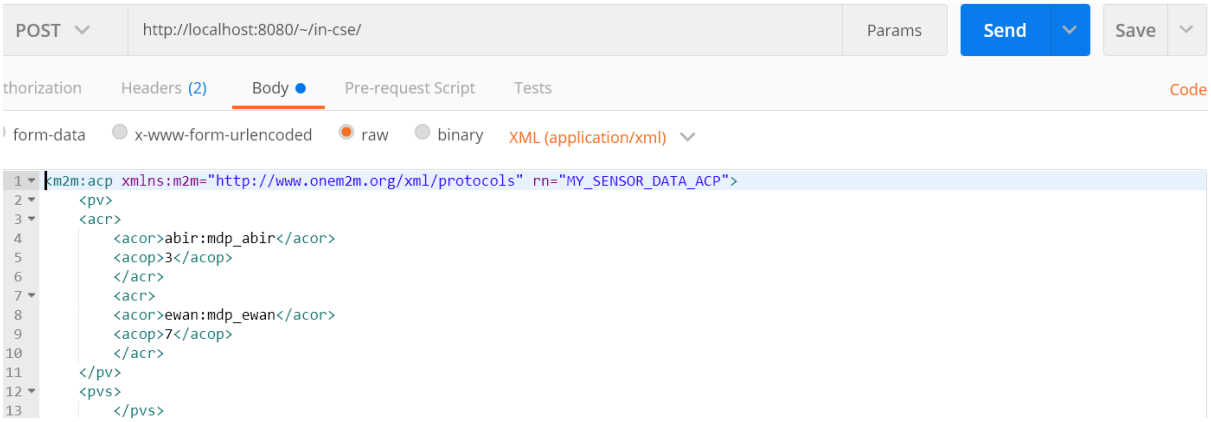
- The privilege tag (pv) determines the access rights of users to the entities linked to the ACP resource.
- The self-privilege tag (pvs) determines the access rights of users on the ACP resource itself.

To determine the access level of a user, the acop parameter is assigned a value equivalent to the sum of the values of the desired access rights. If the user does not exist, it is created. The following table corresponds to the values and their corresponding rights, the sum is 63.

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

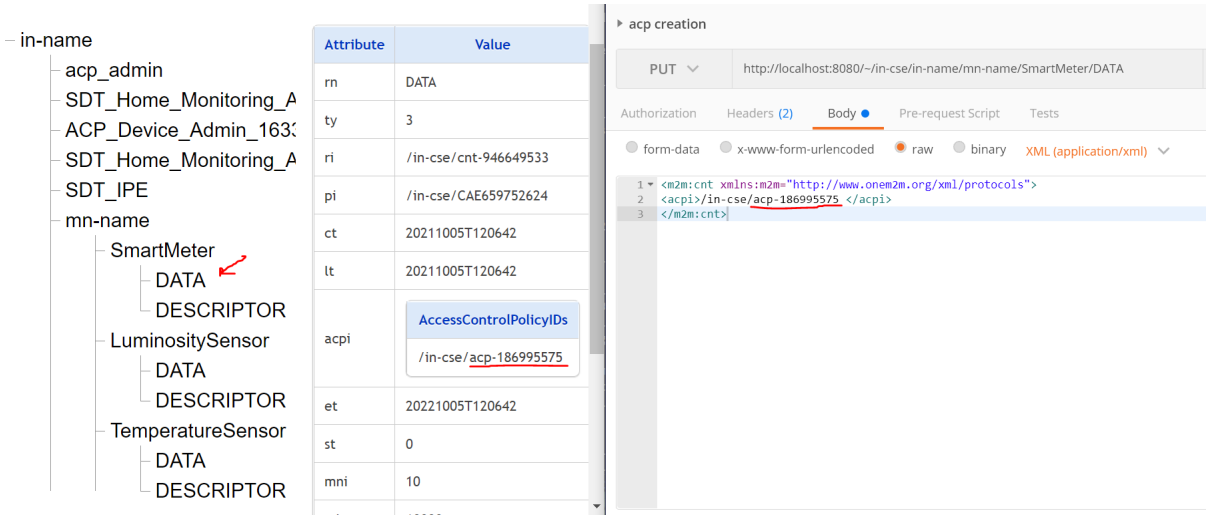
For instance, if a rule is aimed to provide rights to perform RETRIEVE + DISCOVERY + NOTIFY operations, the value of the parameter will be 2 + 16 + 32 = 50.

We will now create access rights for users Abir and Ewan. Below is the creation of the ACP ressource.

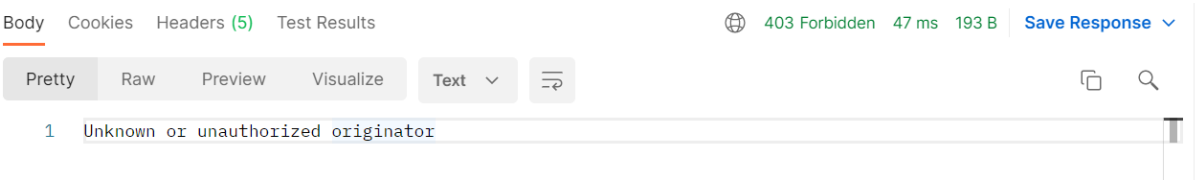


Here Ewan has an acop parameter of 7 : 1+2+4 : CREATE RETRIEVE UPDATE.
User Abir only has an acop of 3 : 1+2 : CREATE RETRIEVE.
We must now link the ACP ressource to what we actually want to protect!

Using the PUT request, we will modify the acpi tag in the body of the container we want to link with the ACP resource. To do this, we have noted in the tree the value acp-186995575 which corresponds to the address of our previously created resource : it is the field we will modify in the put request.



Our container is now linked to the new ACP rules we have set, and the acpi field in the data container now contains two acp addresses. If we try any action with the wrong user profile, this will result in an error/denial response.



Due to lack of time, we were not able to complete the Scenario part of the practical.

Conclusion

This practical has allowed us to have a basic understanding of the REST API OneM2M architecture. Thanks to these exercises, we now know how to manage the resource architecture, create content Instances pertaining to specific containers themselves pertaining to specific application entities. We were also introduced to how to manage different rights.