Rapport TP1: OneM2M REST API

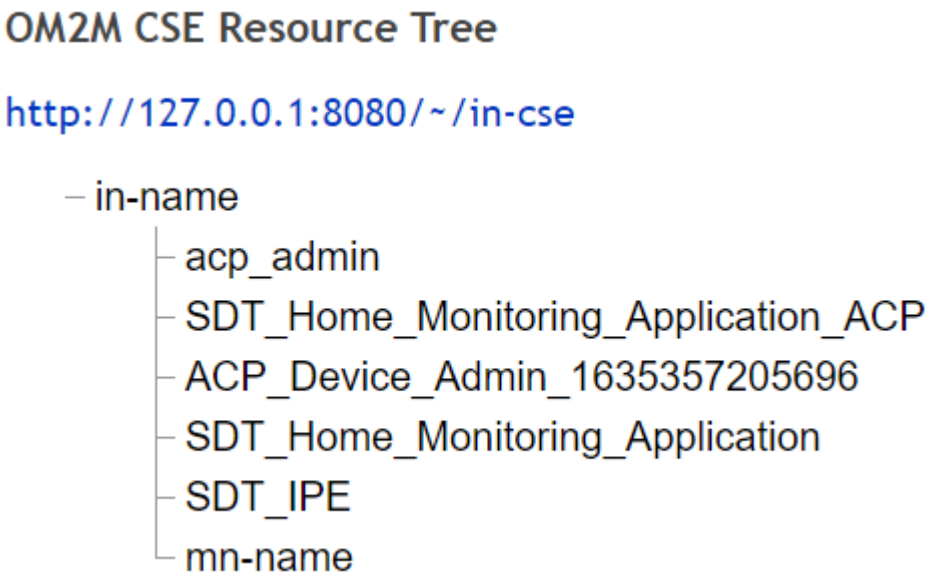
Grégoire HEBRAS
Rami KARAOUD

Démarrage de la plateforme et de l'interface développeur:

Pour ce TP nous avons besoin d'instancier deux nodes, un Infrastructure node et un Middle node. Ici, les deux nodes seront lancés en local.
Nous travaillons sur Windows, nous devons donc exécuter le script "start.bat" ("start.sh" sur Linux) présent dans le répertoire in-cse. Une fois lancé nous avons accès à une console osgi, on peut notamment voir un résumé des plugins actifs grâce à la commande "ss".

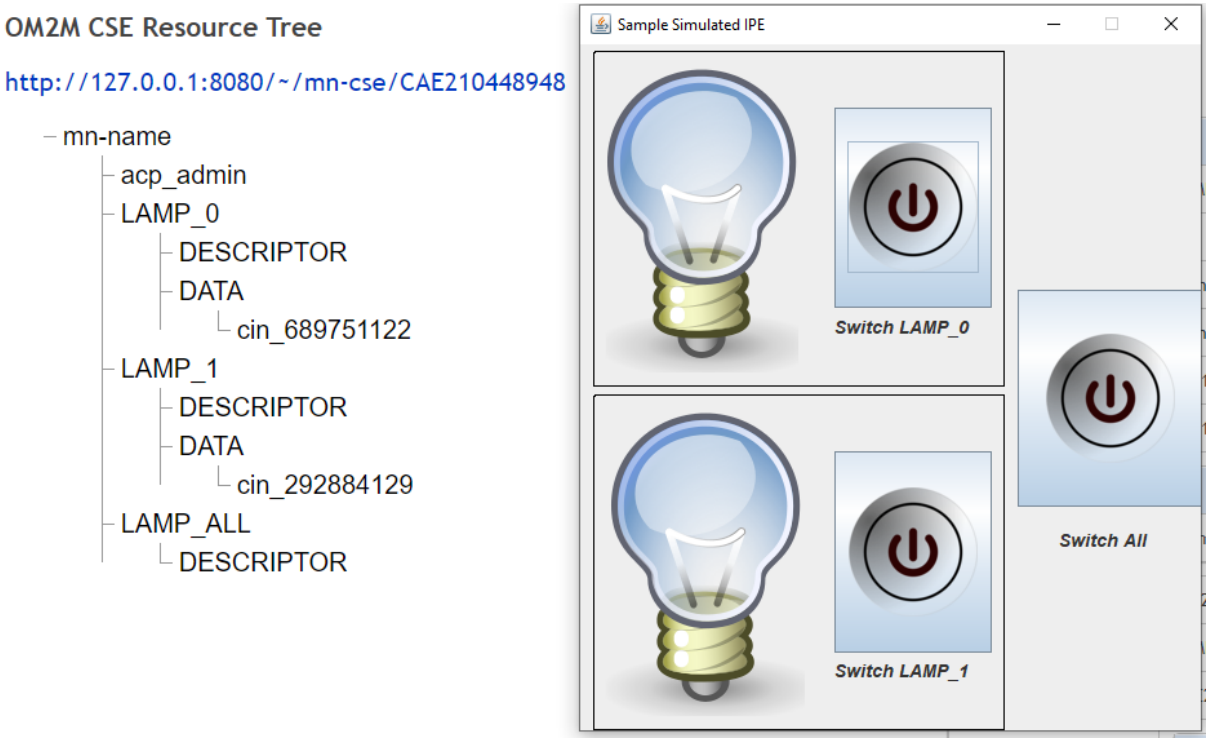
Nous pouvons ensuite accéder à l'interface web développeur IN-CSE, pour cela on se connecte à une adresse de la forme:
http://<ip>:<port>/webpage
Dans notre cas, en local: "http://127.0.0.1:8080/webpage"
On se connecte en tant qu'administrateur avec les identifiants par défaut: admin:admin
Depuis l'interface web on à accès à l'arbre de ressource.

De la même façon que pour le IN-CSE on lance le middle node en local avec le script "start.bat". L'infrastructure node étant déjà lancée, le middle node s'y connecte automatiquement. Sinon, il envoie des requêtes d'authentification toutes les dix secondes.
On remarque sur l'interface web la présence du node mn-cse dans l'arbre du in-cse, preuve de la connexion réussie entre les deux nodes.

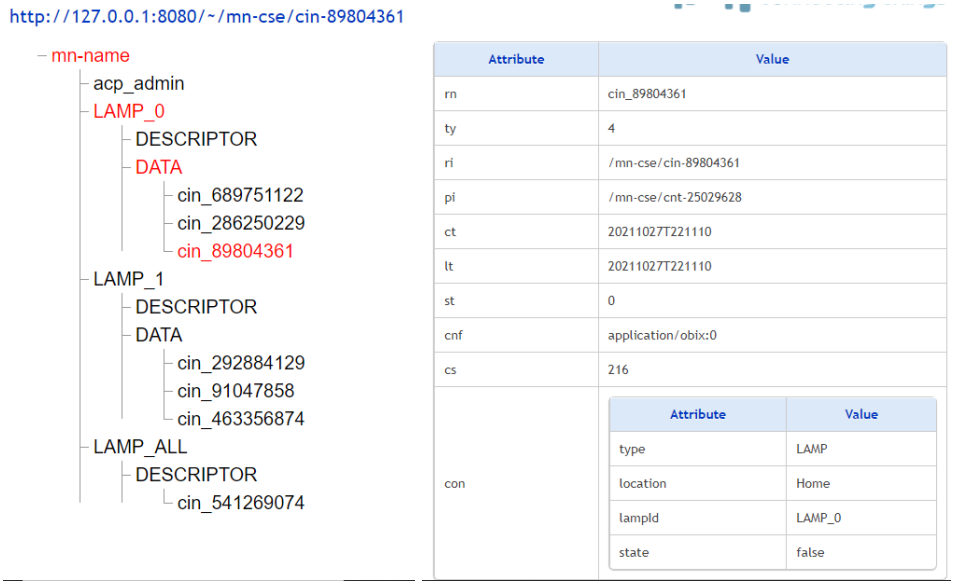


Exemple d’utilisation de l’interface développeur:

Nous allons maintenant démarrer le plugin ipe sample. On peut voir son état et son id grâce à la commande “ss” dans la console osgi, et on la lance avec la commande “start <id>“. En l'occurrence l’id de ipe sample est 41.
Une fenêtre java est générée, il y a deux ampoules (correspondant à des AE) et trois boutons. Pour chaque AE est associé deux containers dans l’arbre des ressources: DESCRIPTOR et DATA.



Cliquer sur les boutons “Switch” créer un content instance dans la branche DATA de la lampe correspondante, et ce pour chaque changement d’état. De plus, en cliquant sur une content instance ou peut lire le nouvel état de l’AE en question à la ligne “state”.



Interaction avec REST API:

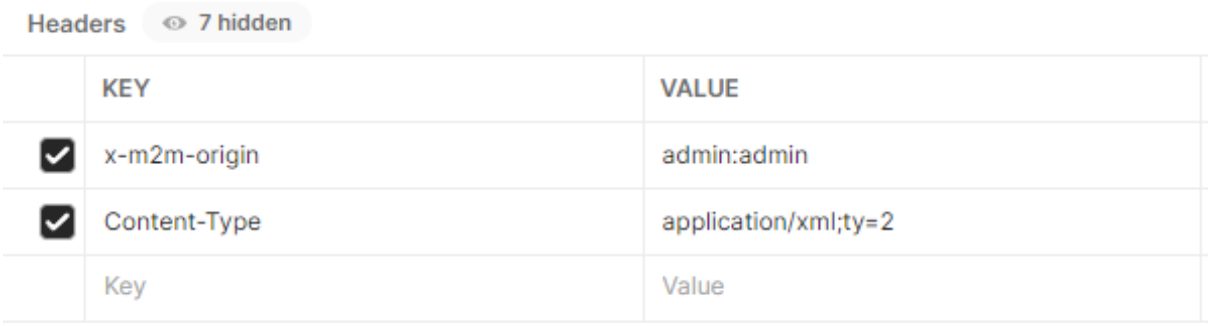
Pour interagir avec le REST API on utilise un REST Client permettant de générer des requêtes HTML. Ici on utilise Postman.
Postman nous permet par exemple de créer des AE, des Containers (CNT) ou des Content Instance (CIN) en utilisant les requêtes ci-dessous. Il est nécessaire d'ajouter des headers:

- x-m2m-origin : admin:admin

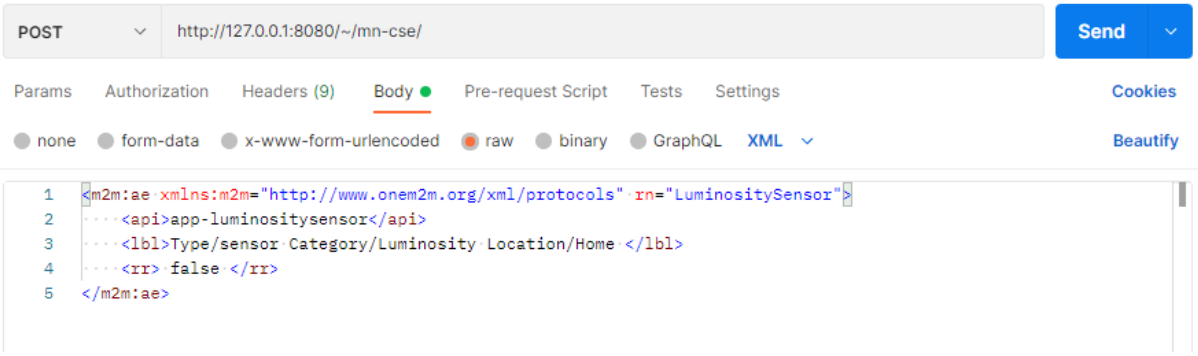
Ce header est spécifique à oneM2M, il permet à l'auteur de la requête d'être reconnu.
On uilise aussi:

- Content-Type : application/xml;ty=2 (or 3, 4,...)

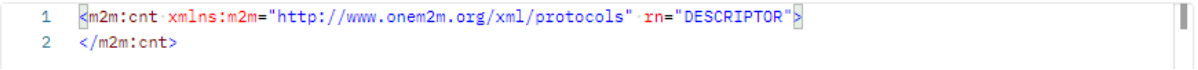
"ty" varie en fonction du niveau auquel on créer des objets: 2pour un AE, 3 pour un CNT, 4 pour un CIN



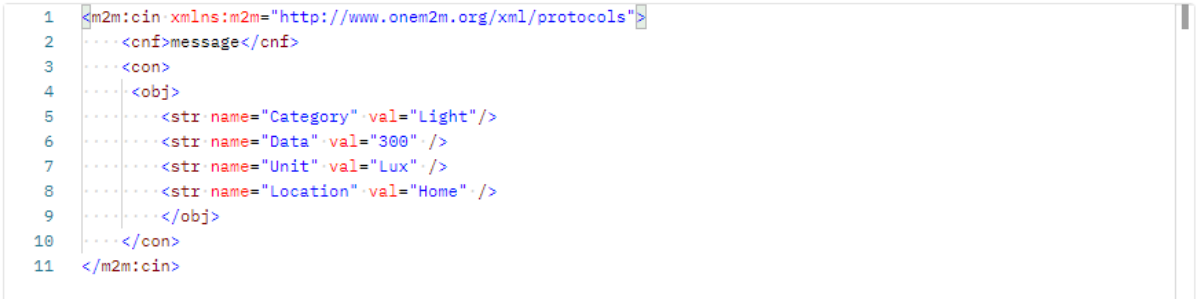
Le code écrit dans le body pour la création d'un AE:



création d'un CNT:



création d'un CIN:



Access Control Management in oneM2M

Avec oneM2M on peut gérer les droits d'accès d'un utilisateur aux ressources, pour cela on utilise les ressources ACP (Access Control Policy), elles représentent les droits accordés à une entité. cette resources a deux attributs spécifiques:
"privileges" (<pv>) correspond au droit d'accès à une ressource liée à l'ACP.
"self-privileges" (<pvs>) correspond au droit d'accès à la ressource ACP elle-même
Ces attributs ont chacun une liste de règles ("rules") portant les attributs suivant:
"Access Control Originator" (<acor>): liste des utilisateurs à qui la règle est appliqué
"Access Control Operation" (<acop>): liste des opérations permises à l'utilisateur en question.

Pour donner un exemple, un AE aura un ACP listé dans son attribut "acpi" (pour 'acp ids'), les règles de privilèges listé dans cet ACP seront appliquées aux utilisateurs qui tentent d'accéder à l'AE.

les opérations accessible par un utilisateur donné sont définies par un entier défini par la somme des valeurs correspondantes au actions autorisées (cf tableau ci dessous)

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

Exercice:

Dans cet exercice nous allons créer une ressource ACP et la lier à l'AE "LuminositySensor"

POST http://localhost:8080/~/mn-cse/mn-name

Params Authorization Headers (10) Body Pre-request Script Tests Settings

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

```
<m2m:acp xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="LuminositySensor_ACP">
    <pv>
        <acr>
            <acor>gregoire:hebras</acor>
            <acop>2</acop>
        </acr>
        <acr>
            <acor>rami:karaoud</acor>
            <acop>11</acop>
        </acr>
    </pv>
    <pvs>
        <acr>
            <acor>admin:admin</acor>
            <acop>63</acop>
        </acr>
    </pvs>
</m2m:acp>
```

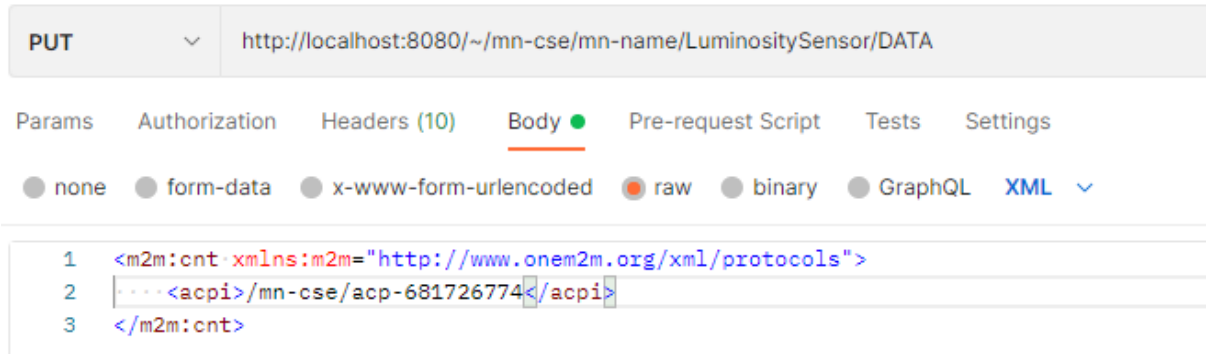
ICI on créer la ressource ACP. l'utilisateur "gregoire" ne peut que retrieve des données (monitor application) et l'utilisateur "rami" peut retrieve create et delete. A titre d'exemple, une "sensor application" classique prendrait la valeur 3 afin de pouvoir create et retrieve. Nous avons ajouter une entité admin capable de tout faire.

http://localhost:8080/~/mn-cse/acp-681726774

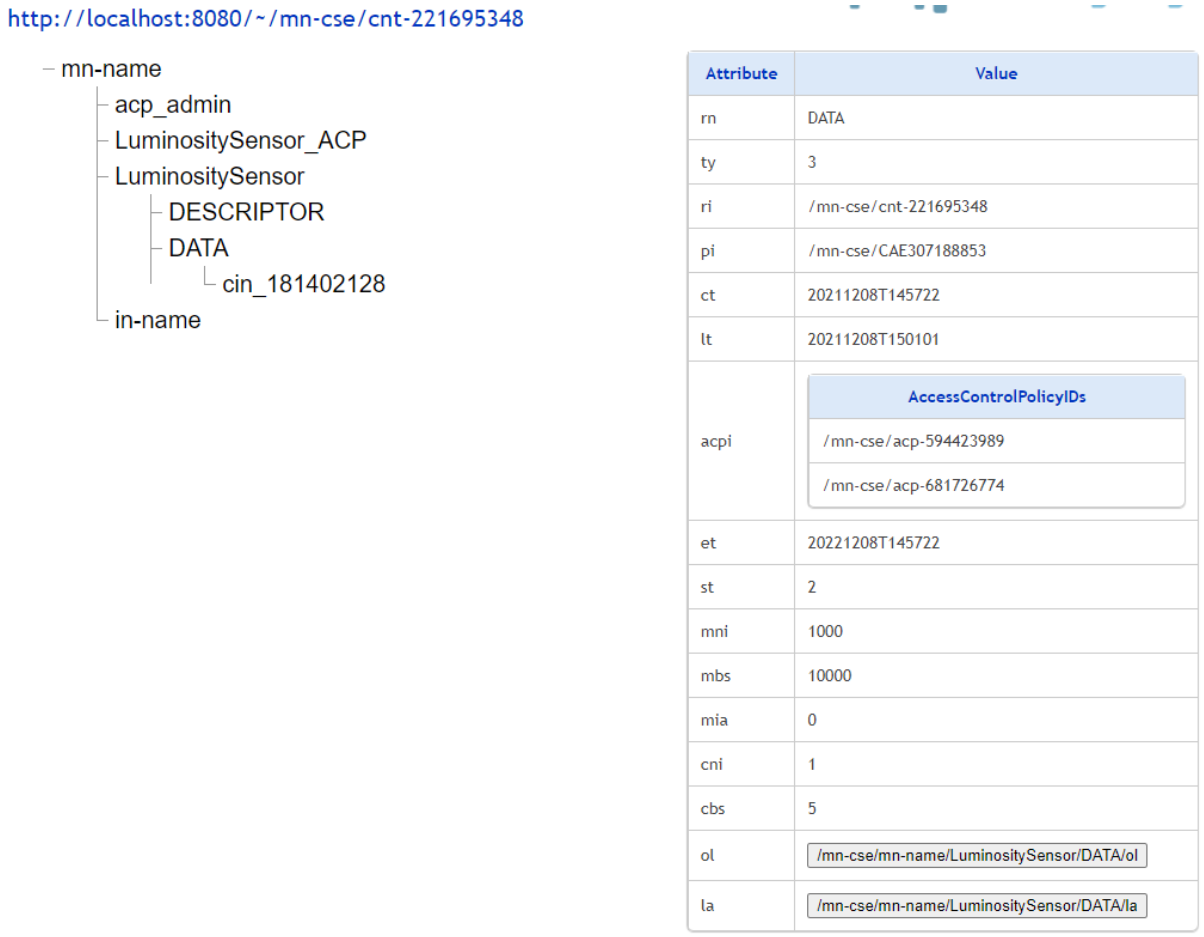
- mn-name
  - acp_admin
  - LuminositySensor_ACP
  - LuminositySensor
    - DESCRIPTOR
    - DATA
      - cin_181402128
  - in-name

Attribute	Value
rn	LuminositySensor_ACP
ty	1
ri	/mn-cse/acp-681726774
pi	/mn-cse
ct	20211208T145912
lt	20211208T145912
pv	AccessControlOriginator: gregoire:hebras — AccessControlOperation: 2 AccessControlOriginator: rami:karaoud — AccessControlOperation: 11
pvs	AccessControlOriginator: admin:admin — AccessControlOperation: 63

On lie ensuite cette ressource ACP à l'AE LuminositySensor.



Pour cela on a récupérer la ressource id de la ressource ACP: acp-681726774



La ressource ACP est maintenant visible dans les acpi de l'AE LuminositySensor.

Nous n'avons malheureusement pas eu le temps en séance de TP de faire la partie scénario portant sur le déploiement d'une flotte de capteurs et d'une application monitor utilisant un mécanisme de souscription et de notification.

Conclusion:

Ce TP nous a permis de découvrir une architecture REST API OneM2M, nous savons maintenant comment créer ce genre d'architecture, gérer des content instances et attribuer des droits d'accès grâce au resources ACP.